UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

TREBALL FINAL DE GRAU

SISTEMA DE TEST PER A EQUIPS DE DETECCIÓ DE TOCATS A L'ESPORT DE L'ESGRIMA

(TEST SYSTEM FOR TOUCH DETECTION EQUIPMENT IN THE SPORT OF FENCING)

Grau en Enginyeria de Sistemes TIC

Autor: Marc Garcia Colomé

Director: Francesc Xavier Moncunill Geniz

Curs: 2015/2016

Localitat: Manresa

RESUM DEL PROJECTE

L'esgrima té una llarga tradició des del seus orígens on s'utilitzava en duels per resoldre disputes. En el món modern, s'ha convertit en un esport refinat que es concentra en la tàctica i l'habilitat de pensar ràpid sota pressió. L'esport ha format part de les olimpíades des dels seus inicis i ha anat evolucionant per afegir els diferents estils de sabre, floret i espasa. En les competicions modernes, al tractar-se d'un esport de precisió, s'utilitzen equips que ajuden a detectar els tocats per facilitar la feina als àrbitres a l'hora de decidir qui és el guanyador. Recentment estan proliferant els sistemes de detecció sense fils que incorporen millores tecnològiques per fer que els esportistes puguin practicar l'esport amb més comoditat.

L'objectiu del present projecte ha estat dissenyar un sistema capaç de posar sota test equips destinats a la detecció de tocats en l'esport de l'esgrima, tant els que utilitzen cables com els que funcionen sense fils. Aquest sistema pot ser utilitzat per mesurar els temps dels tocats i verificar si compleixen les normatives per poder ser homologats. Sens dubte, disposar d'un aparell d'aquestes característiques serà una eina útil per al desenvolupament de nous equips de detecció de tocats.

S'ha realitzat un prototipus demostratiu que incorpora uns actuadors que simulen els tocats dels tiradors realitzats en un combat, uns sensors de llum que recullen les indicacions que dóna el sistema sota test, i un sistema microcontrolador basat en la plataforma Arduino per processar tots els senyals. El sistema inclou connectors estàndard per als mòduls dels tiradors i una connexió USB per comunicar l'Arduino amb un ordinador, des d'on s'enviaran les instruccions per als tests. Per facilitar la realització dels diferents tests, s'ha dissenyat una interfície gràfica des d'on l'usuari pot introduir múltiples paràmetres d'intervals de temps i modalitats per personalitzar les proves tenint en compte les seves necessitats. Finalment, el sistema s'ha posat en pràctica per comprovar el seu correcte funcionament utilitzant un equipament de detecció de tocats sense fils. El sistema ha sigut capaç de detectar els temps de resposta per als tocats simples així com tocats dobles (empats) amb èxit i ha pogut determinar el marge d'error d'un equip en desenvolupament.

ABSTRACT

The fencing has a long tradition from its origins in duels where it was used as a means of settling disputes. In the modern world, it has become a refined sport that focuses tactics and the ability to think fast under pressure. The sport has been part of the Olympics since its inception and has evolved to add different styles such as saber, foil and epee. In modern competitions, being a sport of precision, some equipment is used to help detecting the touches to make it easier for referees to decide who the winner is. Recently some wireless detection systems are proliferating that incorporate technological improvements so that the athletes can practice the sport with more comfort.

The aim of this project was to design a system capable of putting under test the equipment used to detect touches in the sport of fencing, for those using wireless and the ones that still work with wires. Such system can be used to measure the times of touches and verify if they meet the standards to be approved. Undoubtedly, having a device like this will be a useful tool for the development of new detection equipment.

A demonstrative prototype has been made that incorporates actuators that simulate the touches of the competitors that occur in combat, light sensors that collect the information given by the system under test, and a microcontroller system based on the Arduino platform for the processing of all the signals. The system includes connectors for standard modules and a USB connection to communicate the Arduino with a computer, from where the instructions will be sent for tests. To facilitate the accomplishment of different tests, a graphical interface has been designed from where the user can introduce multiple parameters of time ranges and modalities to personalize the different tests depending on his needs. Finally, the system has been put in practice to verify its proper functioning using wireless touch detection equipment. The system has been able to detect the response times for single touches as well as double touches (tie) successfully and has been able to determine the error range of an equipment in development.

ÍNDEX

ÍNDEX DE FIGURES

1. INTRODUCCIÓ

1.1. Introducció a l'esgrima

L'esgrima és un esport de combat en el que dos contrincants, anomenats *tiradors*, s'enfronten per intentar tocar-se utilitzant una arma blanca i aconseguir punts al tocar al contrincant en una zona vàlida. Els tiradors van degudament protegits i l'arma té forma d'estoc (amb un fil sense afilar) amb tres estils diferents en funció del qual se'n diferencien les modalitats: sabre, espasa i floret.

L'esgrima ha existit des que es van voler desenvolupar els combats amb espasa per duels i la defensa personal. Originari d'Espanya, on es van escriure els primers manuals a finals del segle XIX, l'esport es va anat refinant al llarg dels anys a Itàlia i el Regne Unit fins arribar a una versió millorada a França d'on es va adoptar la terminologia actual. L'esgrima forma part de les Olimpíades modernes des del seu inici als jocs d'Atenes del 1896 i al llargs dels anys ha anat evolucionant i se li ha anat afegint les tres modalitats fins arribar a l'estat actual.

Inicialment en les Olimpíades s'utilitzaven quatre àrbitres per regular els combats i assignar els punts als tiradors. Tractant-se d'un esport de gran precisió i per tant, molt difícil de valorar amb l'ull humà, amb el pas del temps es va incloure el sistema de cables elèctrics (1933) que senyala els tocats amb un so i una llum verda o vermella, reduint així el nombre d'àrbitres i aconseguint una valoració més precisa i justa.

No obstant, el sistema elèctric implica elements físics com cables que poden influir negativament en la comoditat dels jugadors a la hora de moure's amb total llibertat. L'evolució de la tecnologia ha permès que sigui possible retirar els cables i en la última dècada s'han dissenyat sistemes per fer la detecció de tocats sense fils.

1.2. Antecedents

Aquest projecte segueix una evolució directa d'un seguit de treballs anteriors que han sigut necessaris per desenvolupar un sistema de detecció de tocat sense fils i que es poden agrupar tenint en compte un seguit de fases.

En una primera fase es va crear el sistema des de zero i es van implementar les funcionalitats més bàsiques. Dos projectes realitzats alhora van fer possible aquesta primera fase: "Ingeniería completa de un sistema wireless para la detección de tocados de esgrima: Sensor." (COSTA – UPC. 2009) [1] i "Ingeniería completa de un sistema wireless para la detección de tocados de esgrima: Sistemas microcontrolador y de

comunicaciones." (COSTA – UPC. 2009) [2]. El sistema resultant va ser un prototip funcional anomenat Wireless Fencing del que se'n va arribar a demanar una patent. Tot i que no es va considerar prou exitós per arribar-se a comercialitzar serviria de base pels projectes següents.

En la segona fase va servir per millorar la tecnologia, pràcticament redissenyant-la des de zero amb la intenció de afegir-li noves funcionalitats, fer-la més fiable i robusta i reduir-ne el consum i el cost així com la mida final del producte. Hi van haver tres projectes que es van realitzar paral·lelament per dur a terme totes aquestes millores: "Diseño y optimización de un sistema de detección de tocados inalámbrico para el deporte de la esgrima: Módulo estático." (SOLANS – UPC. 2011) [3], "Diseño y optimización de un sistema de detección de tocados inalámbrico para el deporte de la esgrima: Módulo dinámico." (PLA – UPC. 2011) [5] i "Protocolo de comunicaciones para un sistema de detección de tocados en esgrima basado en el estándar IEEE 802.15.4" (SOLANS – UPC. 2011) [4].

El producte final va ser un sistema amb el mòdul central i els dels tiradors redissenyats i un software totalment nou que van fer el sistema més modern i amb característiques més atractives per la seva comercialització.

La tercera fase va millorar el sistema per fent-lo més intuïtiu per al usuari afegint missatges durant el combat i més indicacions lumíniques i acústiques per facilitar la configuració dels aparells i senyalitzar més tipus de tocats. Per afegir aquestes funcionalitats i per assegurar-se que complia uns requisits per ser homologat es va tornar a millorar tant el hardware i software. El projecte que va complir amb aquests objectius va ser "Incorporación de un algoritmo de supervisión y otras mejoras a un sistema de detección de tocados inalámbrico para el deporte de la esgrima" (ROYO i PEÑA – UPC. 2011) [6].

Al llarg del temps les necessitats van anar canviant i es demanaven uns nous requisits, per tant, va ser necessari revisar de nou el software dels mòduls i el protocol de comunicacions per afegir-li la capacitat de capturar i filtrar interferències com altres transmissions en l'espectre del protocol ieee 802.15.4 o soroll. El projecte que va solucionar això va ser "Protocolo de asignación dinámica de canal para la comunicación de un sistema de esgrima inalámbrico basado en el estándar ieee 802.15.4" (FERREIRO – UPC. 2016) [7].

El resultat d'aquests projectes és un producte anomenat *Wireless Fencing* que compleix les funcionalitats necessàries per dur a terme un combat d'esgrima i que en teoria compleix tots els requisits. Però la detecció de tocats sense fils és una tecnologia molt recent que s'utilitza des de fa pocs anys i amb la que encara s'està experimentant en les olimpíades. Al no ser totalment infal·lible els requisits del sistema per ser utilitzats oficialment varien sovint i d'aquí sorgeix la necessitat d'un sistema que sigui capaç de

testejar aquests altres sistemes per comprovar mitjançant diferents proves que es compleixen els temps de resposta homologats que permetrien verificar el producte com a vàlid per al seu ús oficial, i amb aquest propòsit, neix aquest projecte.

1.3. Estat de l'art

Actualment al mercat ja existeixen alguns sistemes de detecció de tocats que funcionen sense fils i tants altres que estan en ple desenvolupament. Tot seguit s'enumeren alguns dels més coneguts:

Wireless Fencing

Aquest és el sistema resultant dels projectes anteriors i es pot considerar en desenvolupament ja que encara no s'ha pogut homologar. Està preparat per la modalitat d'espasa i utilitza dos mòduls de tiradors de butxaca, tal i com s'aprecia en la Figura 1, i un mòdul central on es mostren els resultats a través de LEDs.

Les simulacions realitzades al llarg del projecte s'han realitzat mitjançant aquest equipament i ha sigut clau en la realització del projecte.

Figura 1. Wireless Fencing connectat a espases.

Epée Hitmate

Fabricat per Allstar International una famosa marca en el món de l'esgrima. El Epée Hitmate [9] és un producte destinat exclusivament al entrenament, degut a que actualment no compleix oficialment els requisits necessaris especificats per la FIE per ser homologats. De moment només serveix per la detecció en la modalitat d'espasa.

Al igual que el Wireless Fencing utilitza dos mòduls transmissors i un de receptor i indica els tocats mitjançant LEDs verd i vermell en el receptor.

WF1

Fabricat per Favero Electronics és un sistema bastant complert que detecta tocats en les modalitats d'espasa i floret i compleix les especificacions de temps de la FIE. També inclou modes específic per armes de plàstic amb connexió elèctrica.

Utilitza dos terminals per als tiradors i un gran mòdul central amb una sèrie de LEDs verd i vermells per indicar els resultats.

1.4. Objectius

Aquest projecte recull el testimoni de projectes anteriors realitzats per la UPC que ha dedicat anys en el desenvolupament d'un prototip de detecció de tocats. El següent pas és la creació d'un sistema que pogués validar que aquests prototips compleixen un seguit de normes. Els objectius que s'han marcat per aquest projecte s'enumeren a continuació:

- Dissenyar un sistema que simuli tocats d'espasa (sense necessitat d'aquesta), individualment per cada jugador, en diferents intervals de temps.
- També ha de ser capaç de simular tocats simultanis dels dos tiradors però amb una lleugera diferencia per assegurar els temps en que es determina l'empat o un guanyador.
- Tots aquests càlculs han d'estar disponibles en les tres modalitats diferents d'esgrima: espasa, sabre i floret.
- Ser capaç de detectar les respostes del sistema sota test per interpretar-ne els resultats.
- El sistema ha de proporcionar una interfície gràfica perquè l'usuari pugui especificar els paràmetres de les simulacions amb comoditat i senzillesa.
- Utilitzar el sistema fabricat per comprovar si el prototip sense fils dels projectes anteriors compleix les normatives imposades per la FIE.

1.5. Fases del projecte

Per entendre una mica millor l'evolució del treball a continuació es descriuen breument les fases en les que s'ha dividit el projecte.

- Inicialment, s'ha fet un estudi de les diferents maneres de aconseguir els objectius per estimar quins components de hardware serien necessaris i com es farien les seves connexions.

- En segon lloc, a partir d'una placa protoboard, s'ha muntat a gran escala el disseny de hardware fet, per comprovar el seu funcionament un cop connectat amb un Arduino. El muntatge realitzat en la protoboard simulava el sistema Wireless Fencing mitjançant uns LED's ja que inicialment no es disposava d'ell.
- Utilitzant el microprocessador de l'Arduino s'ha programat per realitzar els tests per calcular els temps de resposta exigides per la homologació de la FIE quan es tracta dels diferents tipus de tocat.
- Al disposar del Wireless Fencing s'han dut a terme les simulacions per comprovar el seu correcte funcionament i descobrir possibles complicacions que han aparegut al implementar-lo.
- Amb els tests funcionant correctament s'ha començat a programar una interfície gràfica per a l'usuari utilitzant wxPython per poder realitzar els tests amb comoditat i claredat, sota les ordres dels usuaris.
- Amb la part del programari acabada, s'ha realitzat una placa de circuit imprès on comprimir el hardware per obtenir un producte més petit, pràctic, i robust.
- Finalment s'han realitzat diferents proves amb el producte final i el Wireless Fencing per així verificar el funcionament de la placa.

2. REQUERIMENTS DEL SISTEMA

El propòsit principal d'aquest projecte és verificar que l'equipament d'esgrima compleix un seguit de regulacions establertes, en concret uns temps determinats per la Federació Internacional d'Esgrima (FIE). La FIE és la organització encarregada de l'esgrima olímpica, de regular les normes a nivell competitiu i per tant, la que determina les normatives sobre la sensibilitat i els límits de temps, que l'equipament ha de complir per passar els criteris de homologació amb els que es treballaran en aquest projecte. A continuació es llisten els últims valors actualitzats per la FIE en desembre de 2015 [11].

2.1. Especificacions dels temps per homologació

2.1.1. Espasa

Tocat simple: El sistema ha d'estar tancat (tocat per espasa) durant 6 ± 4 milisegons [11](p.84). Per un senyal de durada t milisegons:

- $t < 2ms$: En cap cas ha de senyalar tocat. Si ho senyala, el sistema falla el test.
- $2ms \leq t \leq 10ms$: Qualsevol indicació es considera correcta.
- $t > 10ms$: Sempre ha de senyalar tocat. Si no ho senyala, falla el test.

Tocat doble: El tocat d'un tirador ha de bloquejar al de l'altre als 45 ± 5 milisegons [11](p.84). Sent t la diferencia en milisegons entre l'inici del tocat del primer tirador i l'inici del segon:

- $t < 40ms$: Si senyala doble tocat el test és correcte.
- $40ms \leq t \leq 50ms$: Qualsevol indicació es considera correcta.
- $t > 50ms$: Mai ha de senyalar doble tocat. Si ho senyala, falla el test.

2.1.2. Sabre

Tocat simple: El sistema ha d'estar tancat com a mínim entre 0.1 i 1 milisegons [11](p.87). Per un senyal de durada t milisegons:

- $t < 0.1ms$: En cap cas ha de senyalar tocat. Si ho senyala, el sistema falla el test.
- $0.1ms \leq t \leq 1ms$: Qualsevol indicació es considera correcta.
- $t > 1ms$: Sempre ha de senyalar tocat. Si no ho senyala, falla el test.

Tocat doble: El tocat d'un tirador ha de bloquejar al de l'altre als 170 ± 10 milisegons [11](p.86). Sent t la diferencia en milisegons entre l'inici del tocat del primer tirador i l'inici del segon:

- $t < 110ms$: Si senyala doble tocat el test és correcte.
- $160ms \leq t \leq 180ms$: Qualsevol indicació es considera correcta.
- $t > 180ms$: Mai ha de senyalar doble tocat. Si ho senyala, falla el test.

2.1.3. Floret

Tocat simple: El sistema ha d'estar tancat durant 14 ± 1 milisegons [11](p.81). Per un senyal de durada t milisegons:

- $t < 13ms$: En cap cas ha de senyalar tocat. Si ho senyala, el sistema falla el test.
- $13ms \leq t \leq 15ms$: Qualsevol indicació es considera correcta.
- $t > 15ms$: Sempre ha de senyalar tocat. Si no ho senyala, falla el test.

Tocat doble: El tocat d'un tirador ha de bloquejar al de l'altre als 300 ± 25 milisegons [11](p.82). Sent t la diferencia en milisegons entre l'inici del tocat del primer tirador i l'inici del segon:

- $t < 275ms$: Si senyala doble tocat el test és correcte.
- $275 \leq t \leq 325ms$: Qualsevol indicació es considera correcta.
- $t > 325ms$: Mai ha de senyalar doble tocat. Si ho senyala, falla el test.

En la següent taula s'aprecien de manera resumida els requisits explicats. Tots els temps estan en milisegons.

	SIMPLE		DOBLE	
	Min (ms)	Max (ms)	Min (ms)	Max (ms)
Espasa	2	10	40	50
Sabre	0.1	1	160	180
Floret	13	15	275	325

Figura 2. Resum de requeriment del temps pels tocats en milisegons.

Tenint en compte que el temps requerit més petit de tots és de l'orde de 100µs en el cas del sabre, s'intentarà que el sistema pugui respondre amb la suficient rapidesa.

2.2. Especificacions d'impedàncies

Aquestes regles per als temps són aplicables tant per l'equipament sense fils, com aquells que necessiten una connexió física. No obstant, la normativa de la FIE també especifica uns valors de resistència per l'equipament.

2.2.1. Floret

El tocat s'ha de registrar si la resistència del circuit és superior a 500Ω, però no ha de fer si és inferior a 200Ω. Un toc a la pista o a la guàrdia s'hauria d'enregistrar si la resistència entre la guarda/pista i la jaqueta és inferior a 100Ω i no ho hauria de fer si és superior a 150Ω.

2.2.2. Espasa

S'ha d'enregistrar el tocat si la resistència del circuit és inferior a 100Ω, però no ho ha de fer si la resistència amb pista o guàrdia és inferior a 100Ω.

2.2.3. Sabre

S'ha de detectar un tocat si hi ha una resistència externa inferior a 100Ω. Un tocat a la guàrdia o la pista haurà de registrar-se si la resistència entre la guàrdia i jaqueta és inferior a 250Ω.

Aquests valors per les resistències s'haurien de tenir en compte en el cas de l'equipament amb fils, però no té sentit quan es parla d'un sistema sense fils, ja que no conté un circuit tancat. Per poder calcular aquestes impedàncies s'hauria de fer un sistema individual per cada tipus de equipament amb i sense fils.

Tenint en compte que aquest treball pretén ser útil per als dos tipus de sistema, es centrarà només en els càlculs dels temps per a l'homologació.

2.3. Separació de masses

Un altre aspecte d'importància, és aconseguir que la massa del sistema que es vol dissenyar, i el del l'equipament a testejar siguin independents per dues raons:

- Per seguretat; és a dir, es vol que els dos sistemes estiguin aïllats elèctricament. En el cas que es connectés un sistema de tocats amb fils, si aquest treballa amb tensions més altes que el que s'està construint podria afectar-lo negativament i malmetre'l.

- Respectar el mode de funcionament dels sistemes sense fils, on s'aconsegueix la detecció dels tocats amb un node de referència propi per cada mòdul de tirador.

2.4. Connexions

Per permetre que el producte final pugui connectar amb l'equipament que s'intenta posar sota test, s'ha de tenir en compte el tipus de connector que utilitza el mòdul de tirador.

Per connectar l'arma a aquest mòdul es fa servir un cable amb iguals connectors mascle als extrems. Aquest connectors consten de 3 pins banana, dos d'ells separats 1.5 cm i el tercer 2 cm tal i com s'aprecia en la Figura 3. Per tant, és necessari que es disposi d'un connector femella per aquesta connexió.

Figura 3. Cablejat arma-mòdul tirador.

3. DISSENY DEL HARDWARE

A continuació s'expliquen quins components de hardware seran necessaris per al muntatge del sistema, les seves característiques i el disseny de la placa de circuit imprès, no obstant, per fer-ho abans és necessari entendre com funcionen els sistemes de tocats sense fils per als que s'està fent aquest projecte, en concret s'explicarà el Wireless Fencing ja que es tindrà accés a ell i s'hi treballarà en la majoria del projecte.

3.1. Funcionament dels sistemes de detecció de tocats

3.1.1. Sistemes sense fils

El treball de (FERREIRO – UPC. 2016) [7] va concloure amb un sistema de tocats per espases al que s'ha tingut accés i que s'ha utilitzat al llarg d'aquest projecte. El sistema consta de 3 mòduls; dos mòduls de tiradors que porten cada un dels participants a sobre, que s'encarreguen de capturar els tocats i enviar-ne la informació al tercer mòdul que farà les funcions de centre de control (mòdul central) per determinar quin participant ha fet el tocat o si hi ha hagut un empat en cas d'un doble tocat.

Els mòduls dels tirador consten de 3 terminals cada un: A, B i C. En el terminal C s'utilitza un generador de senyal alterna que està connectat a un node de referencia independent per cada mòdul. Els terminals A i B capturen tocats vàlids quan es produeix un curtcircuit entre ells generat a la punta de l'espasa i no es detecta aquesta alterna. Si el curtcircuit es produeix quan la punta de l'arma entra en contacte amb la cassoleta del terminal C o la pista, es detecta l'alterna i es considera un tocat no-vàlid.

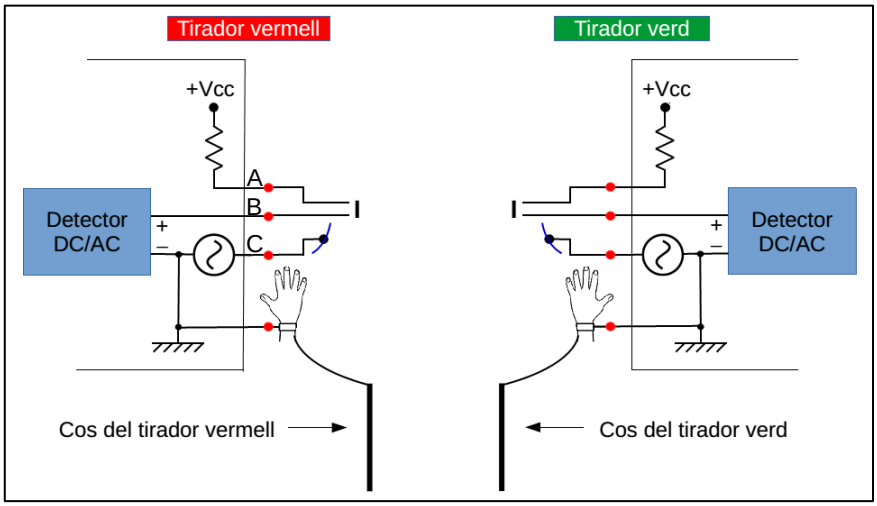

Figura 4. Esquema de funcionament del *Wireless Fencing.*

Les comunicacions amb el mòdul central es realitzen sense fils i és aquest el que s'encarrega de senyalitzar el guanyador il·luminant diferents LEDs de colors vermell i verd com s'aprecia en la Figura 5.

Figura 5. Mòdul central Wireless Fencing amb LEDs il·luminats.

3.1.2. Sistemes amb fils

Ja que es pretén que el sistema funcioni també per l'equipament que utilitza cablejat, a continuació s'explica com funcionen les tres modalitats. Els mòduls dels tiradors tenen igualment 3 entrades; A, B i C i enregistren el tocat quan el port B rep un voltatge alt (Vcc).

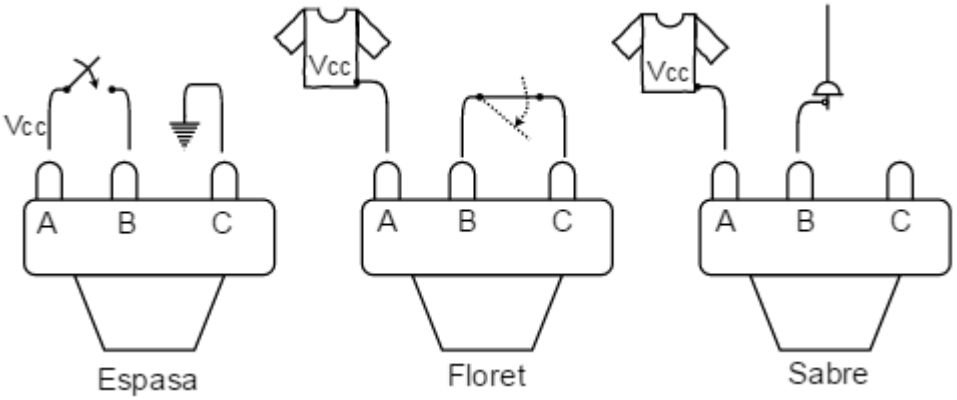

Figura 6. Funcionament de l'equipament amb cablejat.

En el cas de l'espasa, quan es realitza un tocat es crea un curtcircuit entre A i B aconseguint un voltatge Vcc en B. El port C és una massa compartida entre els tiradors.

En l'estat de repòs del floret, quan no s'està fent un tocat, hi ha un curtcircuit entre B i C de manera que el voltatge de B és de 0V. En el moment que es fa un tocat vàlid amb la jaqueta de l'adversari, el curtcircuit queda obert i el terminal B llegeix els 5V als que està alimentada la jaqueta.

Finalment, el sabre utilitza un pull-down per mantenir el port B amb 0V quan l'espasa no fa contacte. En el moment que es realitza un tocat amb la jaqueta, al igual que en el floret, el port B llegeix un voltatge alt.

En tots tres casos es pot simular un tocat si enviem directament un voltatge de 5V pel port B, per tant, tenint en compte els objectius del projecte es treballarà amb els ports A i B, ja que per als càlculs dels temps no és necessari tenir en compte els tocats no-vàlids. Si es volguessin considerar també els no vàlids, a l'igual que en el cas de les impedàncies, s'hauria de fer dos circuits específics per als sistemes amb cablejat i per als que funcionen sense fils, ja que els tocats no-vàlids funcionen de manera totalment diferents en cada cas. Així doncs, el sistema tindrà almenys quatre connexions amb l'exterior (dos ports per mòdul de tirador). El tercer port dedicat a la massa es deixarà flotant i no s'utilitzarà.

3.1. Arquitectura del sistema

En la Figura 7 s'ha dividit el funcionament del sistema en 4 blocs essencials. Els senyals que simulen els tocats s'originaran en el microcontrolador que els enviarà als actuadors. Aquests crearan un curtcircuit entre els dos terminals per on es connecten els mòduls dels tiradors que formen part del sistema a testejar.

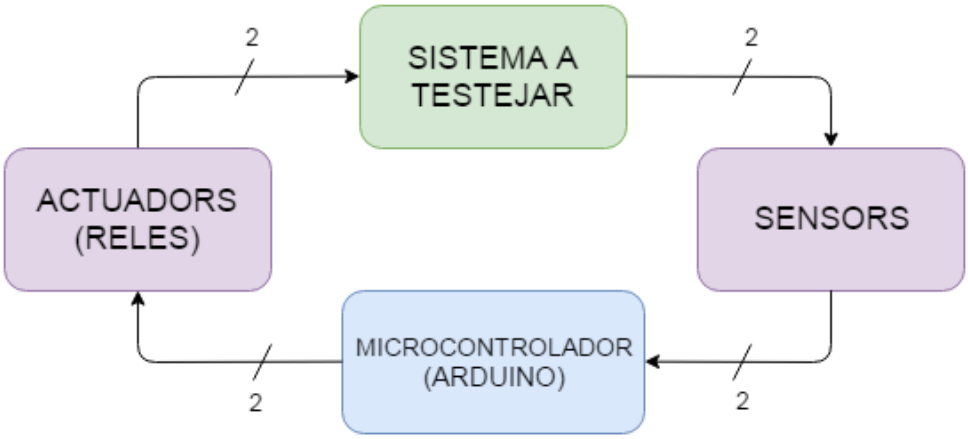

Figura 7. Diagrama de blocs del sistema.

El sistema entén aquest curtcircuit com un tocat, i depenent de la seva durada enviarà un tipus de resposta al mòdul central a través de cablejat elèctric o sense fils depenent del sistema.

En la següent Figura 8 es representa un esquema més detallat dels components i el recorregut dels senyals. El sistema a testejar no forma part directament del muntatge per pel projecte, però és un element intrínsec del seu funcionament.

Finalment, els sensors de llum prèviament col·locats davant dels LEDs del mòdul central, capturaran els canvis de llum que s'enregistraran en el microcontrolador.

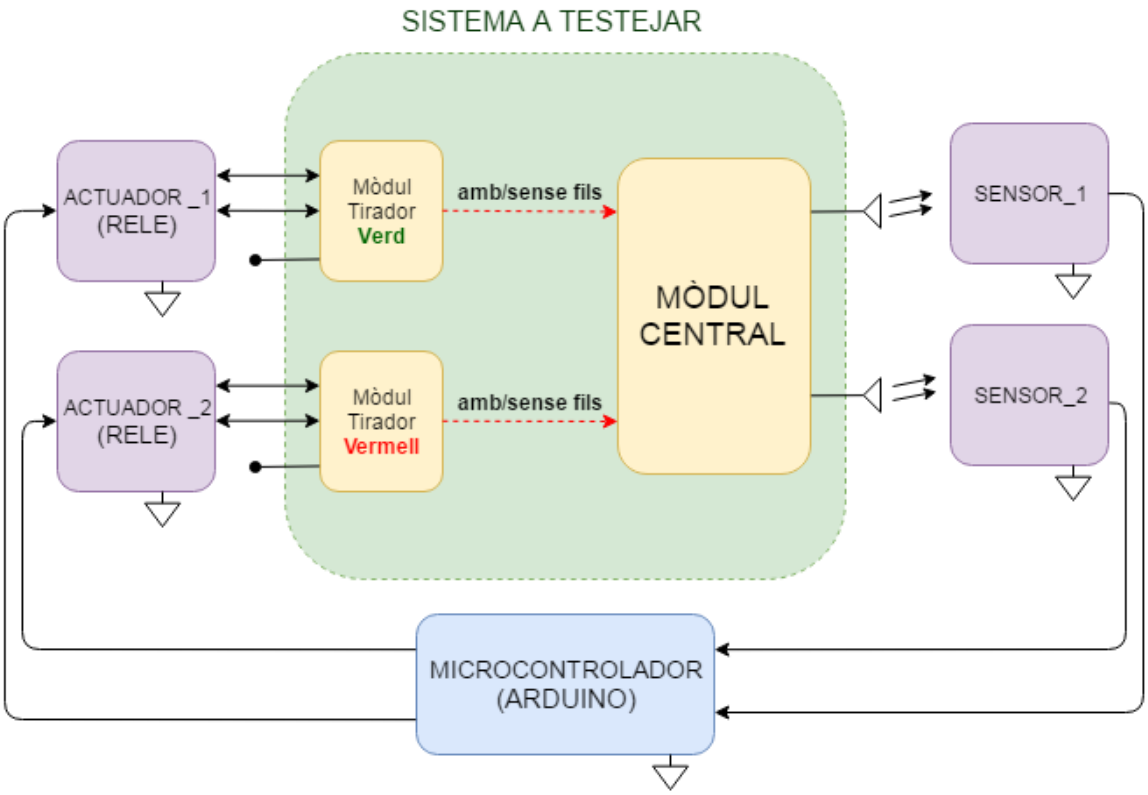

Figura 8. Diagrama de blocs del sistema detallat.

Cal recalcar que la massa del dos sistemes són independents i que la dels mòduls dels tiradors es deixarà flotant. D'aquesta manera s'eviten riscos al connectar un equipament de detecció amb cablejat elèctric que podria tenir una tensió massa elevada i malmetre el nostre sistema i es respecta el funcionament dels equips que treballen sense fils.

3.2. Actuadors

Per solucionar el problema de fer els dos sistemes elèctricament independents s'utilitzaran relés d'estat sòlid. Els relés funcionaran com un interruptor per deixar passar el corrent entre els dos ports del tirador, al crear un curtcircuit, sense que arribi a existir una connexió física entre els dos sistemes. Els relés d'estat sòlid a diferència dels mecànics, permeten canvis en el seu estat a grans velocitats a més a més de evitar el contacte mecànic i, per tant el desgast.

El model ASSR-1218 [13] utilitza un acoblament òptic i té 4 terminals; un primer terminal que s'utilitzarà per enviar el senyal de tocat que encendrà el LED intern, el segon reservat

per la massa d'aquest LED i finalment dos terminals que connectaran amb el mòdul del tirador i que el mantindran elèctricament aïllat gràcies l'opto-acoblament.

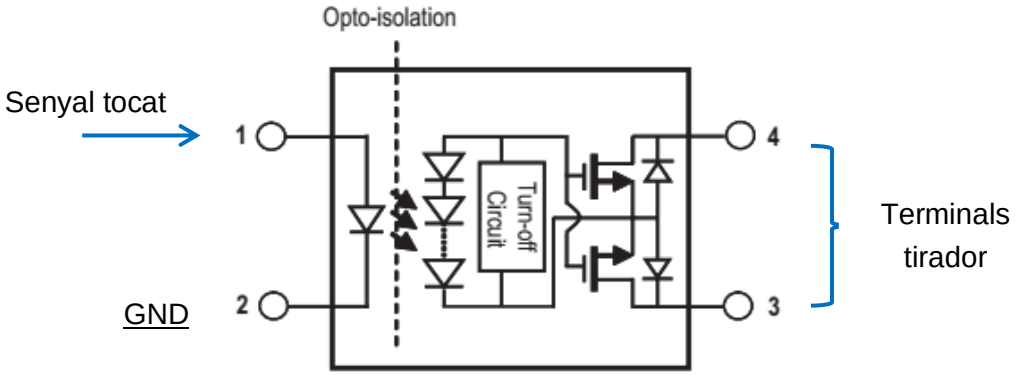

Figura 9. Diagrama elèctric del relé ASSR-1218.

La intensitat d'entrada màxima permesa és de 25mA per al terminal 1 i 200mA per als 3 i 4, mentre que el voltatge màxim per al terminal 1 és de 5V i la tensió entre els terminals 3 i 4 no pot superar els 60V. Els temps d'encesa varien entre 0.7ms i 5ms però els diferents tests realitzats demostren una resposta molt més ràpida i inferior als 0.7ms. Per l'apagada els temps varien de 0.04ms a 5ms i, de nou els tests realitzats mostren una resposta pràcticament immediata, en cap cas proper als 5ms.

3.3. Sensors de llum

En la majoria de sistemes de detecció de tocat, així com el sistema sobre el que s'està treballant, el mètode utilitzat per indicar els diferents tocats és mitjançant uns LED's de colors vermell i verd que representen cada tirador, i per tant al guanyador depenent de quin s'il·lumina. El sistema també emet un fort senyal audible que indica que hi ha hagut un tocat però no distingeix entre tiradors, per tant, la única manera que el sistema pugui diferenciar-los és a través dels LED's.

Per fer-ho s'utilitzaran sensors de llum o fototransistors que es col·locaran propers als LED's i informaran de quan s'han il·luminat. La resposta d'aquests sensors seran les entrades al nostre sistema i al igual que les sortides seran necessàries un mínim de quatre (dos per sensor).

S'ha experimentat amb l'LDR NSL 19M51 [14] i fototransistors model BPX43 [15]. L'avantatge d'utilitzar el fototransistor és tenir una resposta el doble de ràpida, respecte el LDR. No obstant, les dues respostes són suficientment ràpides tenint en compte que un cop un LED s'encén, ho fa durant temps més que el suficient, així que la decisió per escollir un sobre l'altre recau en un aspecte més pràctic.

Alguns dels sistemes de tocat acostumen a tenir el mòdul central muntat en una paret, així que per simplificar les connexions amb el sistema es faran sortir alguns cables de l'encapsulat per evitar necessitar tenir el mòdul central just al costat. Els sensors no formaran part de l'estructura interna del sistema sinó que es podran manipular des de fora per acostar-los als LED's.

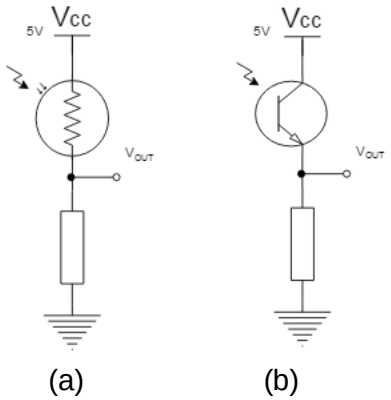

Figura 10. Esquema elèctric del sensors LDR (a) i fototransistor (b).

En el moment de fer la connexió entre el sensor i el LED intervenen masses factors que dificulten la captació correcta de la llum i que donen falsos positius. És important segellar el sensor tan com sigui possible per evitar altre llum que no sigui la del propi LED, com la pròpia llum ambient de l'habitació o un altre LED pròxim al que s'està mesurant.

El fototransistor té una forma cilíndrica i allargada, com es mostra en la Figura 11, que el fa difícil de col·locar de manera que enfoqui correctament amb el LED. El LDR en canvi, al ser petit i pla com una llentia permet una connexió directa a sobre del LED que sempre es troba dins d'un encapsulat llis, utilitzant una cinta adhesiva opaca. Per aquest motiu s'acabarà utilitzant per sobre del fototransistor.

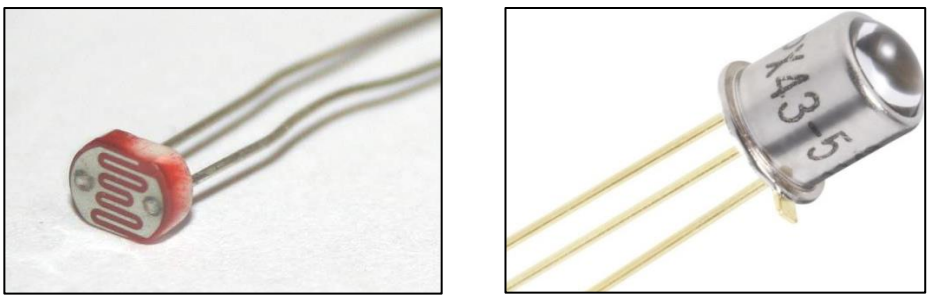

Figura 11. LDR NSL-19M51 a l'esquerra i fototransistor BPX43 a la dreta.

S'han rumiat alternatives al mètode per col·locar el sensor sobre el LED com pinces o marcs de plàstic per fixar-se en una posició, el problema és que cada sistema de tocats utilitza un mòdul central totalment diferent en forma i mida. La falta d'un estàndard en les dimensions fa que sigui complicat adaptar els sensor per tots els casos.

3.4. Microcontrolador

Per acabar, el component més important o en tot cas imprescindible serà un microcontrolador sobre el que programarem totes les funcions que volem que realitzi el sistema i que governarà el comportament d'aquestes entrades i sortides.

L'Arduino UNO és una placa de circuit imprès que ofereix un microcontrolador Atmega328p juntament amb altres perifèrics que compleixen les necessitats per al sistema de test.

A continuació s'especifiquen algunes de les principals característiques:

- Microcontrolador de 8 bits.
- Arquitectura avançada RISC amb 131 instruccions.
- Clock de 16 MHz.
- 32x8 registres de propòsit general.
- 32K Bytes de memòria Flash programable.
- 1K Byte EEPROM i 2KB de SRAM.
- Retenció de dades d'entre 20 i 100 anys.
- 2 Timer/Counters de 8 bits i un de 16 bits amb diferents prescalers i mode de comparació.
- Interfície Mestre/Esclau SPI
- Convertidor analògic-digital de 8 canals de 10 bit
- Serial USART programable.
- Interrupcions en canvis als Pins.
- 6 canals de PWM
- Connector USB per alimentar i establir comunicació amb ordinador.
- 14 pins digitals d'entrada/sortida i 6 analògics d'entrada programables.
- Tensió d'alimentació entre 1,8 i 5,5 V.
- Rang de temperatura entre -40ºC i 85ºC

Veient les especificacions és fàcil deduir que satisfà sobradament els nostres requisits. De totes elles, algunes de les més importants i utilitzades per al sistema són 4 dels pins digitals, el connector USB, les interrupcions en canvis dels pins, una alimentació de 5V i diferents comptadors amb prescaler i mode de comparació.

A més a més de complir les nostres necessitats l'Arduino UNO proporciona altres avantatges:

- El connector USB proporciona una interfície de port sèrie que a més a més de oferir un canal de comunicació entre el microcontrolador i l'ordinador, permet fàcilment afegir o modificar el codi en un futur si l'usuari ho creu necessari. L'Arduino és una placa molt utilitzada i coneguda en el món de la programació i és molt probable que

els usuaris mínimament entesos es sentin còmodes en el moment de remenar el codi.

- Sense necessitat de buscar gaire, es pot aconseguir per un preu considerablement econòmic. Si bé les últimes versions oficials ronden els 23€, es pot trobar al mercat una gran varietat de clons o marques blanques que compleixen exactament amb les mateixes especificacions incloent, dimensions, microcontroladors i fins i tot cables externs. El seu preu varia des dels 2,50 fins els 10 €.
- Comoditat a la hora de programar. Si es té coneixement suficient es pot programar directament el chip Atmega328, però la placa Arduino proporciona una manera molt més còmode simplement utilitzant llenguatge C (com en aquest cas) o utilitzant el programari IDE.

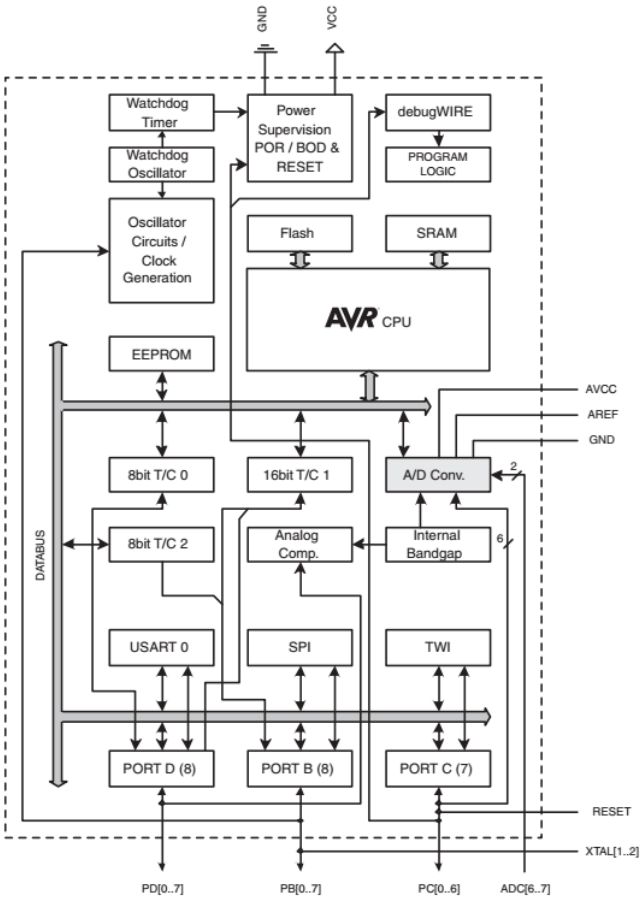

Figura 12. Diagrama de blocs de l'Atmega328p.

3.5. Muntatge experimental

Amb el propòsit de comprovar que el sistema pensat funcioni tal i com s'havia estudiat, i amb el propòsit d'experimentar amb els components del hardware en cas d'errors, inicialment es va realitzar tot el muntatge sobre una protoboard com en la Figura 13.

Al començament no es disposava d'un sistema de detecció de tocats, així que els tocats es simulaven col·locant un LED a la sortida dels relés. Aquest funcionament no representava gaire fidelment el del sistema de tocats però va ser suficient per començar a programar el microcontrolador.

Els sensors de llum estaven encarats directament davant del LED i tapats dins d'un cilindre per limitar la quantitat de llum ambient que pogués entrar.

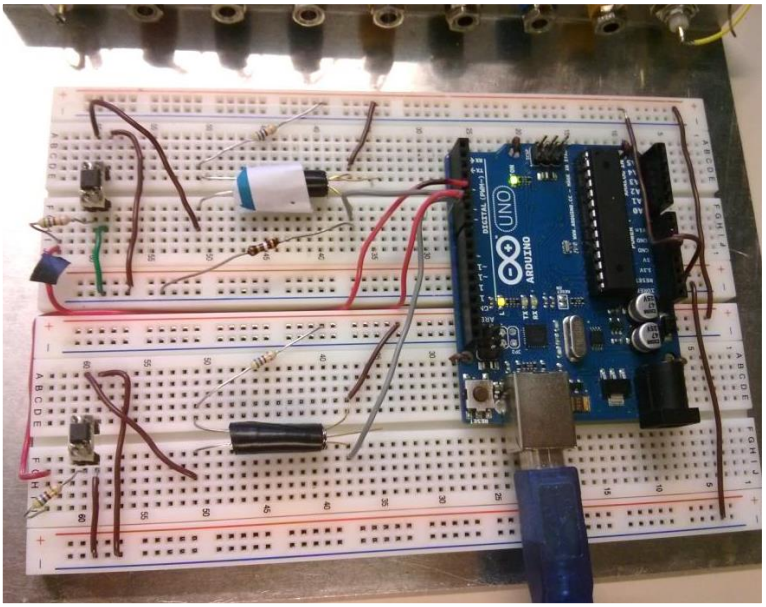

Figura 13. Muntatge experimental en una protoboard.

3.6. Desenvolupament d'un prototipus demostratiu

Per connectar tots els components del hardware es dissenya una placa de circuit imprès (PCB) per connectar-se a sobre de l'Arduino UNO a través dels seus pins. En aquest apartat es detallen els passos seguits per la seva realització.

3.6.1. Esquema elèctric

Primer de tot, en la Figura 14 es mostra l'esquema elèctric dels components que formaran part de la placa de circuit imprès.

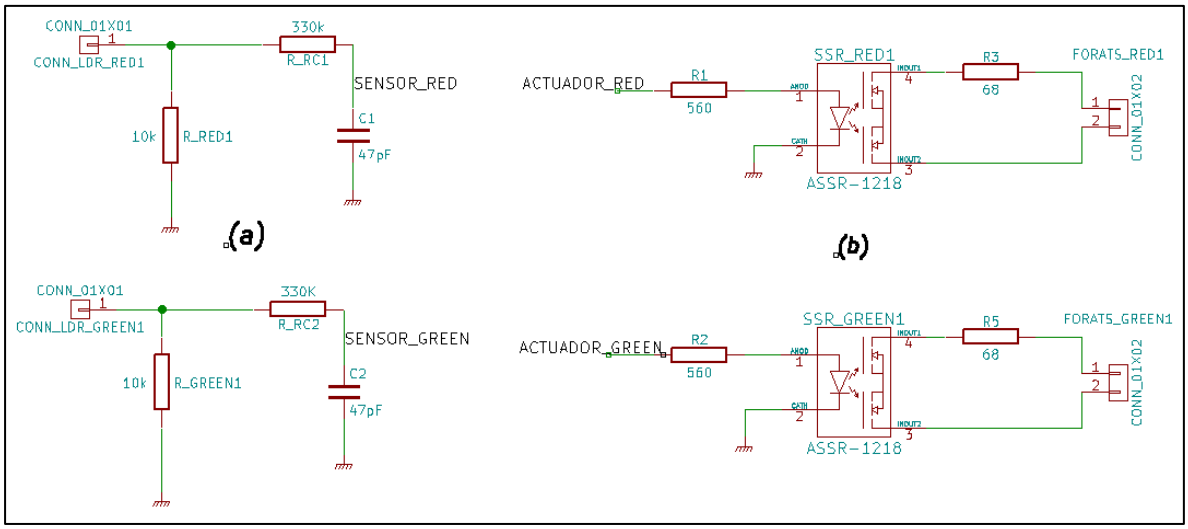

Figura 14. Esquema elèctric de la PCB.

Els elements de la zona (a) són les connexions per als sensors LDR. Un dels terminals del sensor es connectarà a través d'un pin a la placa (CONN_LDR_RED1) mentre que l'altre anirà connectada a un dels pins de la Figura 15 (VCC_LDR_RED1) que l'alimentarà**.** El LDR funciona oferint poca resistència quan està exposat a la llum, i més resistència quan és a les fosques. Per poder determinar el voltatge d'entrada al pin del Arduino s'utilitzarà un divisor de tensió col·locant una resistència entre el LDR i l'entrada al pin. Es vol aconseguir un voltatge tan pròxim com sigui possible a 5V per un LED il·luminat, i tan pròxim a 0V per quan esta apagat.

S'ha experimentat amb la resistència que ofereix el LDR; en el pitjor cas de llum, quan no està perfectament encarat amb el LED, ofereix ~3kΩ, i en el pitjor cas de foscor, quan hi ha llum ambient que es pot filtrar a traves de la carcassa del mòdul central, ofereix 16kΩ. Per tant, utilitzant una resistència intermèdia de 10kΩ s'aconsegueixen 3.3V en els pitjors casos de llum, i 2V en els de foscor.

Tenint en compte que el voltatge necessari perquè el pin digital llegeixi un '1' lògic és a partir dels 2,6V [8](pg.505), amb aquesta resistència s'aconsegueix un voltatge superior per al LED il·luminat i inferior quan esta apagat.

Al mateix node s'hi afegeix un filtre pas baix (RC) que pot servir per eliminar soroll i possibles interferències de fonts externes com per exemple, la il·luminació artificial que produeix fluctuacions de 100Hz (fluorescents). Amb una freqüència de tall de 10Hz:

$$\tau = \frac{1}{2\pi f_c} = \frac{1}{2\pi 10Hz} \approx 16\,ms$$

D'aquesta manera s'aconsegueix atenuar possibles interferències dels fluorescents fins a 10 vegades. Si es considera la resistència del divisor de tensió de 10kΩ, s'escull una R molt més elevada de 330kΩ per al filtre per evitar efecte càrrega.

$$\tau = RC \quad C = \frac{\tau}{R} = \frac{0.016s}{330k\Omega} = 48pF$$

S'escollirà un condensador de 47pF tenint en compte els valors estàndards i per tant, $\tau = 15.5ms$.

Els elements de la l'apartat (b) de la figura són els actuadors (relé). En la sortida del microcontrolador es col·loca una petita resistència per evitar una intensitat superior als 25mA màxims suportats pel relé.

$$R = \frac{V}{I} = \frac{5V}{25mA} > 200\Omega$$

Utilitzant una R de 560Ω s'aconsegueixen 9mA en els terminals d'entrada del relé.

Els ports de la dreta del relé crearan el curtcircuit en la terminal del tirador. En els sistemes de detecció per cables el voltatge dels mòduls dels tirador és aproximadament de 12V, així que es col·loca una resistència per evitar els valors màxim d'entrada per aquests terminals del actuador, és a dir, un voltatge de 60V i una intensitat de 0.20A.

$$R = \frac{V}{I} = \frac{12V}{0.20A} > \mathbf{60}\Omega$$

S'escollirà una resistència de 68Ω tenint en compte els valors estàndard que més s'aproximen i són superiors a 60 i per tant, $I = 0.17A$.

El terminal on es connecta el relé són forats en la placa reservats per a pins on es soldaran cables BANANA per connectar-se amb el mòduls dels tiradors.

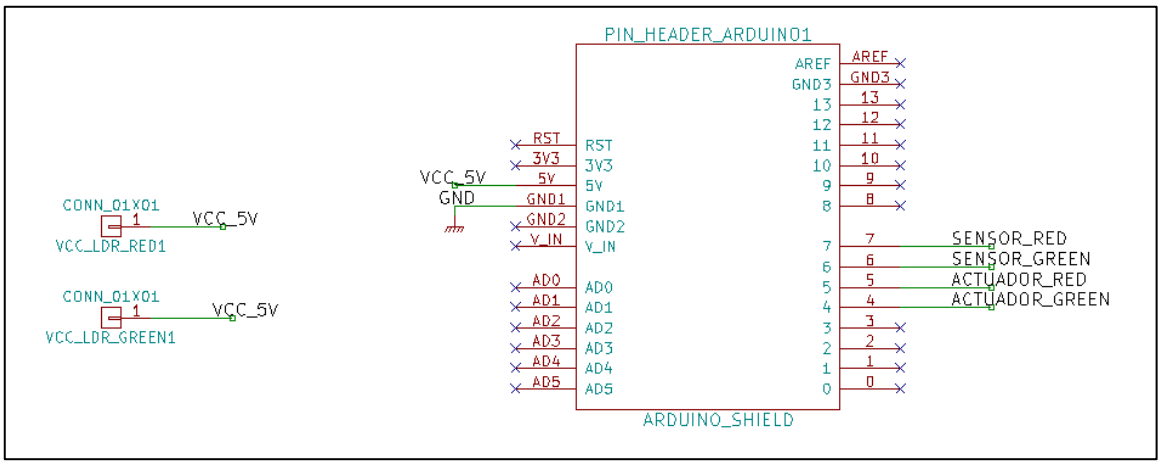

Figura 15. Diagrama elèctric i connexions amb l'Arduino.

En la Figura 15 s'acaba d'apreciar els pins de la placa que serviran d'alimentació per als LDR i la relació entre els nodes esmentats amb els pins corresponents de l'Arduino.

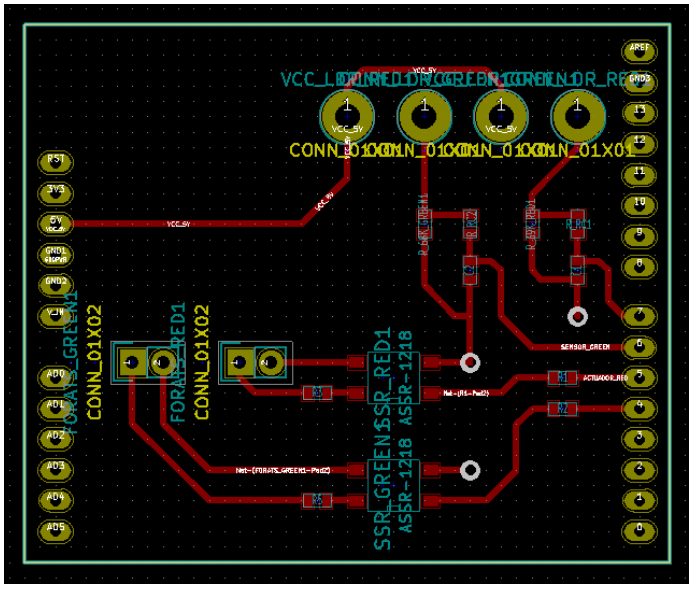

Figura 16. Disseny de la placa de circuit imprès.

La Figura 16 mostra el disseny realitzat amb la distribució del hardware per la placa de circuit imprès. Les dimensions coincideixen en amplada amb les de l'Arduino però no en llargada. Tot i utilitzar nomes 6 dels pins, s'han inclòs un total de 28 potes per connectar amb la majoria de ports de l'Arduino per mantenir la placa inamovible. Els forats reservats per connectar els cables per als mòduls del tirador i els LDR s'han posat uns als costat dels altres deliberadament tenint en compte que sortiran de l'encapsulat per la part inferior de la figura, evitant encreuaments o corbes brusques que castigarien les connexions amb el temps. La pista de GND engloba la majoria de la part inferior de la PCB, tal i com s'observa en la Figura 17, evitant els forats reservats a les potes i connectant amb el pin de massa corresponent de l'Arduino.

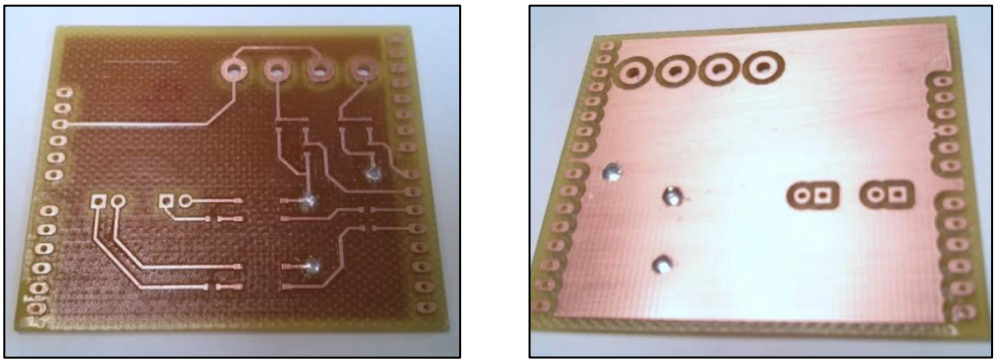

Figura 17. Placa de circuit imprès sense components.

L'últim pas per completar la PCB és soldar la resta de components del hardware així com els pins per les connexions amb l'Arduino i els connectors pels cables externs.

Després d'experimentar amb els components ja soldats, s'ha comprovat que sense el filtratge del circuit RC s'aconsegueixen resultats exitosos, per tant, al final no s'utilitzarà tot i que es deixaran les pistes disponibles per si es vol afegir el filtre en un futur.

3.6.2. Presentació del producte final

El sistema s'ha col·locat dins d'un encapsulat de plàstic dur i robust de 13 cm de llargada, 6,5 cm d'amplada i 2,6cm de profunditat. L'Arduino queda collat en un extrem a través d'uns cargols de plàstic enganxats a la base i unes femelles que es poden retirar amb facilitat en cas de necessitar recuperar-lo. En l'altre extrem s'han perforat 6 forats en la carcassa per col·locar connexions femella per cables BANANA per on es connectaran els mòduls dels tiradors.

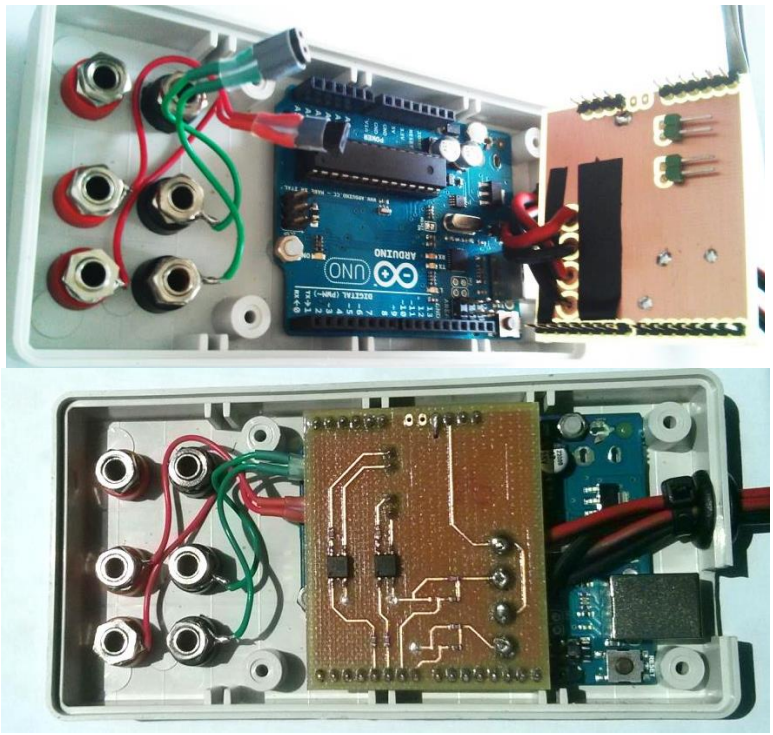

Figura 18. Interior de la caixa amb les connexions.

Tot i que només s'utilitzen dues connexions per mòdul, s'ha afegit un tercer forat que no connecta amb res tenint en compte que el connector dels tiradors té els 3 endolls banana junts en la mateixa peça.

La placa de circuit imprès es pot separar de l'Arduino fàcilment però queda soldada als cables dels LDR. Aquests cables estan units amb una brida i surten a través d'un forat de manera que es puguin manipular els cables sense por de arrencar la soldadura. A l'Arduino se li ha tret el connector Jack per encabir millor en l'espai limitat i, finalment se li ha afegit una ranura al encapsulat per la connexió USB.

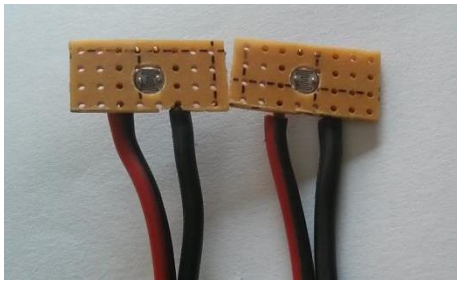

Figura 19. Sensors LDR.

Els LDR estan soldats en una petita placa rectangular amb un forat per fer-hi cabre els sensors de manera que la connexió amb els LEDs resulti més fàcil al fer-se a través de dues superfícies llises.

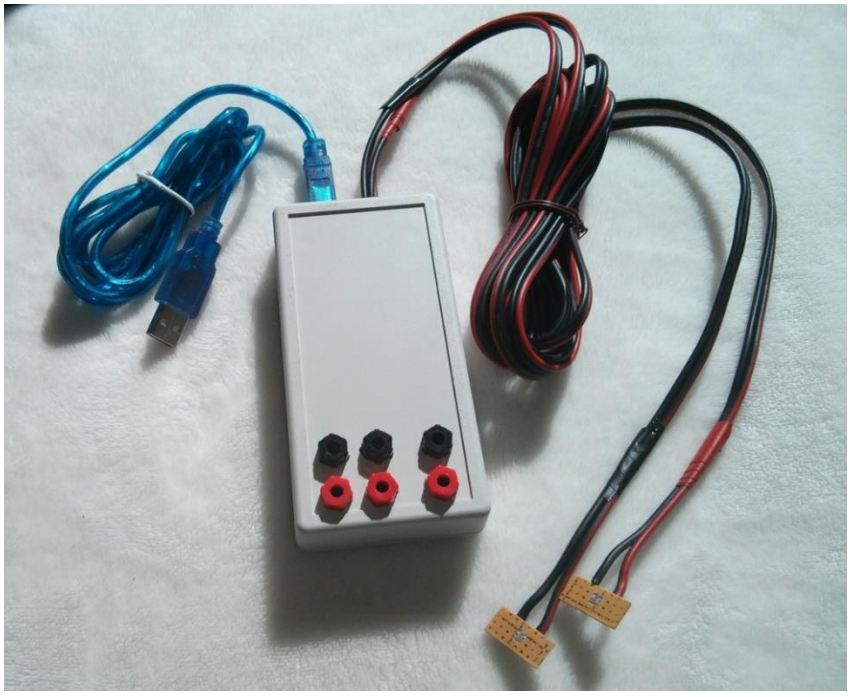

Figura 20. Presentació del prototipus final.

Amb un prototipus demostratiu presentable al públic complet, s'ha decidit anomenar-lo *Fencing Tester*.

3.7. Cost del prototip

A continuació, en la Figura 21 es detallen els preus de tots els components que han sigut necessaris pel muntatge del producte final per aproximar un cost total:

Material	Quantitat	Preu unitari (€)	Preu total (€)
Relé MOSFET ASSR-1218	2u	1.51	3.02
Sensor llum NSL 19M51	2u	0.88	1.76
Arduino UNO	1u	22 (5.02)	22 (5.02)
Carcassa 1599BSGYBAT Hammond	1u	5,29	5.29
Cargol plàstic DURATOOL D00687	2u	0.085	0.17
Rosca M3 HFST Z100	2u	0.0812	0.1624
Resistències SMD	8u	0.0178	0.1068
Connector femella Banana	6u	0.095	0.57
Speaker cable 0.75mm^2	1.5m	0.65	0.975
TOTAL			**34 (17)**

Figura 21. Fitxa del cost dels components.

En la majoria dels casos els components es compren en paquets amb un gran nombre d'unitats. Per aproximar el preu a un valor més real, s'ha tingut en compte el preu proporcional per unitat en aquests casos. La gran majoria dels preus provenen de Farnell [18].

Sense tenir en compte la mà d'obra per muntar tots els elements, el cost final és de 34€. La majoria del cost és degut al Arduino UNO, però com ja s'ha comentat en l'apartat 3.4, existeixen versions de 'marca blanca' amb les mateixes especificacions i el mateix microcontrolador per un preu molt més econòmic a partir de 2.5€. Si l'usuari està d'acord amb aquesta elecció, el preu total amb un Arduino de 5.02€ es redueix considerablement fins a 17€.

4. SOFTWARE DEL SISTEMA

En aquest apartat s'explica tot el funcionament del Fencing Tester des del punt de vista de la programació per entendre a nivell intern com s'està tractant la informació i l'enviament de senyals entre la interfície i el microcontrolador.

4.1. Llenguatges de programació

La programació del treball es pot dividir en dos apartats diferents segons on s'està executant; en l'Arduino o en l'ordinador (interfície gràfica).

En el cas de l'Arduino, o més ben dit del seu microcontrolador, es poden utilitzar diferents eines de programació. Existeix al mercat programari com ArduBlock o Snap4Arduino tot i que estan destinats a un públic més principiant. L'opció més evident seria utilitzar l'Arduino IDE, el software promogut pels propis desenvolupadors de la plataforma Arduino i que compta amb una gran comunitat d'usuaris a internet amb guies, fòrums i ajuda més que suficient. El llenguatge utilitzat es basa en una sèrie de funcions de C/C++.

En aquest cas, s'ha utilitzat precisament llenguatge C, sense cap aplicació addicional, que proporciona funcions i instruccions a més baix nivell que els programaris esmentats. D'aquesta manera es pot tenir un control més precís quan es tracta de accedir a registres i configurar temporitzadors entre altres.

D'altra banda, per al codi que s'executa en l'ordinador per donar instruccions i rebre informació de l'Arduino s'utilitza Python. El llenguatge de Python pot resultar fàcil de programar i entendre, i facilita enormement la comunicació amb l'Arduino, gràcies al mòdul PySerial que dóna accés al port sèrie.

Python també proporciona un gran nombre de paquets diferents per crear interfícies gràfiques com TKinter, PyQt, PyGUI i wxPython. Per al desenvolupament de l'aplicació s'ha utilitzat wxPython que és compatible amb Windows, Unix i Mac.

4.2. Funcionament general

El funcionament entre programes és bastant senzill d'entendre. Inicialment l'usuari executa l'arxiu *fencing.py* per tenir accés a la interfície gràfica. Des d'aquí es poden configurar múltiples paràmetres per als diferents tests que s'envien sota ordre al Arduino.

Cada test inclou diferents durades de temps que es simularan de manera incremental, no obstant, per cada una d'aquestes durades el programa *fencing.py* enviarà la ordre a *main.c* i esperarà la seva resposta abans d'enviar la següent.

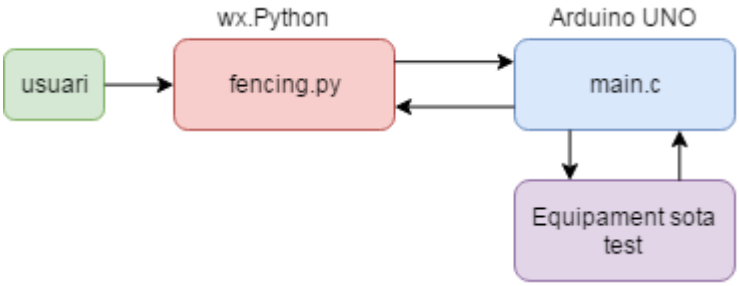

Figura 22. Flux d'informació entre mòduls de programació.

El programa *main.c* s'encarrega d'escoltar pel port sèrie quin tipus de test es vol realitzar i el valor de la durada de temps. Tot seguit, simula el tocat a través de les connexions amb el sistema sota test, i espera la seva resposta. Finalment s'espera aproximadament 4 segons abans de respondre amb el resultat a *fencing.py*.

La raó d'esperar-se tanta estona abans de respondre i realitzar la següent simulació, és per tenir en compte el temps que el sistema sota test utilitza per reiniciar-se i preparar-se de nou després de detectar un tocat. Si s'enviessin les instruccions per la següent simulació abans de finalitzar el reiniciat, no s'obtindria resposta.

4.3. Programació del microcontrolador

En aquest apartat es detallen les característiques més importants de la programació en el microcontrolador, per una vista més detallada d'aquest codi consultar l'Annex 1.

Tal i com s'ha explicat, en el microcontrolador es troba compilat i executant-se indefinidament el programa *main.c*. Les funcions principals d'aquest mòdul són *tocat_simple(), tocat_doble()* i *main().*

4.3.1. Tocat Simple

Aquesta funció té com a paràmetres; el tirador, que pot ser el vermell o verd, i el temps o durada del tocat. Aquesta durada es passa a través d'una variable de 2 bytes (*uint16_t*) i el seu valor representen microsegons amb un mínim de 100µs i un màxim de 25kµs.

La funció s'encarrega de simular un tocat posant un '1' lògic per un port de l'Arduino durant $t\ \mu s$. Per determinar si s'ha detectat el tocat en el mòdul central del sistema sota test, es parteix de la suposició que aquest senyal no s'indica immediatament quan és

rebut o en el moment que el mòdul central determina que té una durada suficient com per considerar-se un tocat vàlid.

Tenint en compte que els temps per la detecció dels tocats dobles són molt més grans que els necessaris per detectar els simples, se suposa que els tocats simples no són confirmats fins que no ha passat prou temps com per descartar un tocat doble. És a dir, si en la modalitat d'espasa es realitzés un tocat de 10ms que el mòdul central detecta com a vàlid, abans de mostrar-ho il·luminant el LED, s'esperarà entre 40ms i 50ms per confirmar que no hi ha un tocat provinent del segon tirador.

Figura 23. Diagrama de temps de tocat simple en espasa.

En la Figura 23 es mostren diferents tocats realitzats un rere l'altre, incrementant la duració en cada iteració (suposant múltiples tests indicats per l'usuari). Tenint en compte que els temps de detecció de doble tocat estipulats per la FIE per la modalitat d'espasa són entre 40ms i 50ms, el sistema probablement no indicarà el tocat vàlid fins al valor màxim dels dos.

En el moment que la funció comença a simular el tocat, s'activen les interrupcions per capturar si s'ha encès o no el LED. Quan el LED s'il·lumina, ho fa durant aproximadament 1 segon, així que les interrupcions es deixen actives durant 350ms, una mica més del temps màxim per la detecció dels tocats dobles en la modalitat de floret (modalitat amb duracions més llargues).

Al finalitzar el test, s'envia un 1 o un 0 a l'aplicació depenent de si s'ha detectat el tocat o no respectivament.

4.3.2. Tocat Doble

Aquesta funció, al igual que en tocat_simple() té com a paràmetres; el tirador i el temps entre tocats. Aquesta durada es passa a través variable de 2 bytes (*uint16_t*) i el seu valor representen milisegons amb un mínim de 1ms i un màxim de 450ms.

El seu funcionament consisteix en simular un tocat per un tirador, i després d'un determinat temps t, simular-ne un altre per al segon tirador. Aquesta diferència de temps entre els dos senyals és el que la FIE exigeix que un tirador ha de bloquejar a l'altre, i és

la variable principal amb la que es cridarà aquesta funció. A diferencia del tocat simple, la resolució per aquest test és de 1ms.

Els dos senyals simulats tenen una durada de 19ms, d'aquesta manera el programa s'assegura que el tocat té una llargada més que suficient per ser detectada per les tres modalitats d'arma, ja que el màxim requerit pel floret, que és el senyal més llarg, és de 15ms.

Al realitzar el càlcul per al tocat doble es parteix de la suposició que el tocat simple funciona correctament, almenys per als casos de tocats de 19ms. Si aquest no fos el cas, es perdria el sentit de realitzar aquest segon test des d'un començament, ja que el sistema seria incapaç de realitzar amb èxit el tocat del primer tirador.

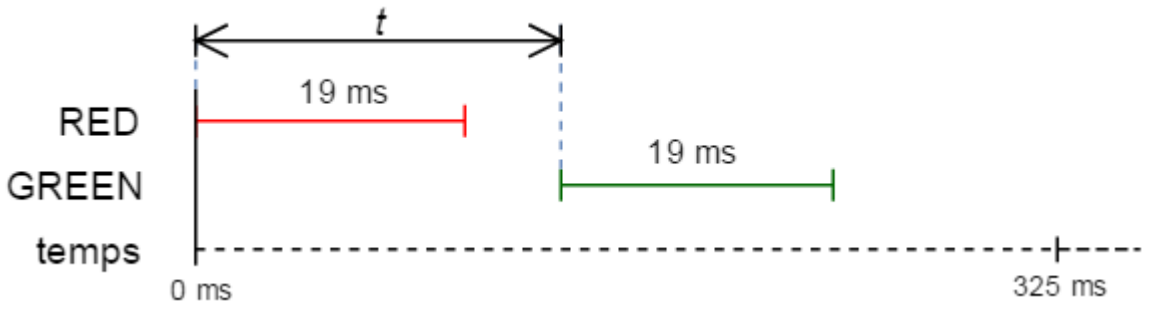

Figura 24. Diagrama de temps de tocat doble en floret.

D'aquesta manera, tal i com s'observa en la Figura 24, es poden realitzar dos tocats que tècnicament no es superposen en el temps, però que l'equipament ha de detectar com a tocat doble si es realitzen dins d'un interval específic de temps. En el cas aquí mostrat, aquest interval finalitzaria als 325ms (floret), moment a partir del qual no ha de considerar un doble tocat si el tirador GREEN inicia aquí el seu tocat ($t > 325ms$) .

Per tenir en compte aquest cas la funció realitza operacions diferents tenint en compte si el temps entre l'inici d'un tocat i el següent ("t" en la Figura 24) és menor o superior de 19ms, és a dir, si s'arriben a superposar els tocats o no.

Al igual que en el tocat simple, añ finalitzar el test, s'envia un 1 o un 0 a l'aplicació depenent de si s'ha detectat un doble tocat o no respectivament. En aquest cas el 0 és indicació de que només s'ha detectat un tocat, no significa que no hagi capturat cap senyal.

4.3.3. Funció main

La funció main() és senzilla però vital, ja que és l'encarregada de mantenir el programa funcionant en un bucle infinit i d'escoltar pel port sèrie si se li demana alguna instrucció.

En el moment que detecta que hi ha informació disponible, llegeix 3 o 4 bytes. El primer byte informa del tipus de test, 'S' pel simple i 'D' pel doble; el segon byte indica el tirador, 'R' pel vermell i 'G' per verd. Per acabar, si el tocat és simple, llegirà 1 byte més amb un valor entre 1-250 que multiplicarà per 100 per convertir a µs, en canvi, si el tocat és doble llegirà 2 bytes per capturar un valor entre 1-350ms.

4.3.4. Configuració de Timers/Contadors

Tal i com s'ha vist, les accions principals del programa es basen en manipular pins durant diferents durades de temps. La eina de l'Atmega328p que ens permet fer-ho són els diferents Timer/Counter.

En el cas del tocat simple, si es considera que la durada mínima a simular és de 100µs, es vol utilitzar un comptador (o *timer*) que necessiti varis cicles per arribar a aquest valor. Amb aquest propòsit s'utilitza un prescaler de 64, d'aquesta manera el clock de l'Arduino de 16 MHz queda dividit suficientment per obtenir una resolució o període de 4µs en el comptador (cada cicle de contat).

En el cas en que la durada del tocat és màxima (25ms), el timer ha de comptar fins a 6250. L'únic *timer* capaç d'emmagatzemar aquest valor és el Timer1 de 16bits i és, per tant, el que s'acaba utilitzant. Aquest Timer1 es configura en mode CTC de manera que, en comptes de contar fins que arribi al valor màxim que pot fer-ho, ho farà fins al valor especificat en OCR1A.

Abans d'introduir el valor al comptador (OCR1A), es resta 1 sabent que el *timer*, quan acabi de comptar, utilitzarà un cicle més per reiniciar-se, així que se li treu el cicle sabent que l'executarà i s'aconseguirà el temps exacte desitjat.

$$\frac{temps\ màxim\ \ a\ comptar}{resolució/període\ \ timer} = \frac{25000\mu s}{4\mu s} = 6250\ cicles \qquad OCR1A = 6250 - 1\ cicles$$

Alternativament, en els tocats dobles s'accepten un mínim de 1ms i un màxim de 450ms, per tant, és necessari utilitzar un prescaler més gran de 256 per dividir el clock i aconseguir un període també superior de 16µs. Per comptar fins a 450ms s'ha de configurar el OCR1A amb un valor de 28125, que segueix sent possible ja que el OCR1A és un registre de 2 bytes.

$$\frac{temps\ màxim\ \ a\ comptar}{resolució/període\ \ timer} = \frac{450ms}{16\mu s} = 28125cicles \qquad OCR1A = 28125 - 1\ cicles$$

En aquest exemple, els cicles del OCR1A són exactes, no obstant, no sempre és així, sovint la divisió dóna decimals. En aquestes ocasions es pot corregir, si al finalitzar el comptador anterior, es canvia de nou el Timer1 amb el prescaler 64(4µs) i es compta el temps restant per aconseguir el valor exacte.

Per acabar, s'utilitza el Timer1 amb un prescaler de 1024 i període de 64µs, per la funció delay_ms() que s'utilitza per afegir els retards de aproximadament 4 segons, necessaris per reiniciar els mòduls dels tiradors entre tocats.

4.4. Programació de la interfície gràfica i mòdul central

En aquest apartat es detallen les característiques més importants de la programació per la interfície gràfica, per una vista més detallada d'aquest codi consultar l'Annex 2.

L'aplicació que s'executa des de l'ordinador i que comunica l'Arduino amb l'usuari està programat utilitzant Python 2.7. Aquest programa es pot considerar el principal ja que s'encarrega de construir una interfície gràfica, enviar els paràmetres de les simulacions al microcontrolador, i processar les respostes per determinar si entren en els paràmetres de la homologació de la FIE. Totes aquestes funcionalitats s'executen des de l'arxiu *fencing.py*.

4.4.1. Entorn gràfic

L'entorn gràfic s'ha construït utilitzant wxPython, una adaptació de la llibreria gràfica wxWidgets. Tant el wxPython com el mòdul PySerial utilitzat per la comunicació amb l'Arduino són compatibles amb Windows, Linux i Mac, així que amb uns petits retocs en la configuració del port de l'Arduino, la interfície es converteix en multiplataforma.

L'aplicació és bastant intuïtiva; ofereix la possibilitat d'escollir entre les tres modalitats d'esgrima, i cada cop que s'escull una d'elles, els paràmetres introduïts per defecte canvien als dels temps necessaris per la homologació.

Hi ha possibilitat de realitzar 2 tipus de tests diferents, tocats simples i dobles, i per cada un d'ells es permet escollir un vector de temps introduint un temps inicial, un temps final i un salt que representa l'increment. Òbviament no s'accepten valors finals superiors als inicials i si amb el salt introduït no s'arriba amb exactitud al temps final, l'últim valor del vector serà l'anterior més pròxim al final de manera semblant a com es fa amb el programari de càlculs matemàtics.

Els tocats simples accepten un inici mínim de 0.1ms i un màxim de 25ms amb una precisió de salt de 0.1ms. Els dobles accepten un inici mínim de 1ms i un màxim de 450ms amb una precisió de salt de 1ms. Aquests valors s'han determinat tenint en compte el rang on treballen els intervals dels temps estipulats per la FIE.

Es poden escollir els tiradors per realitzar els tests de un en un, però si s'escullen múltiples tiradors i tipus de test alhora, l'aplicació els realitzarà tots un darrere l'altre. Tenint en compte que per cada test s'han d'esperar 4 segons pel reiniciat del mòdul de

tirador, al introduir un gran nombre de tests alhora pot mantenir a l'aplicació ocupada una bona estona.

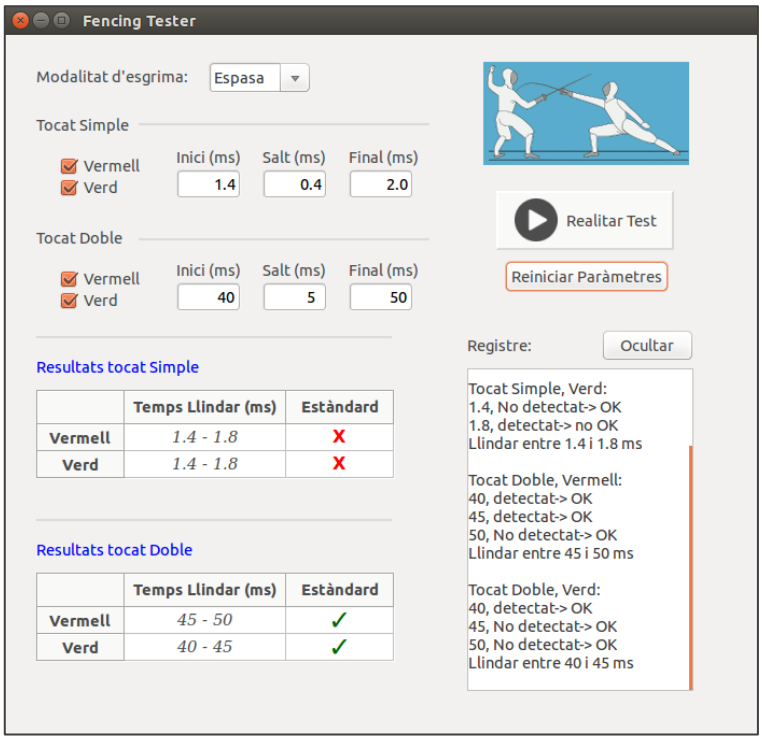

Figura 25. Interfície gràfica creada amb wx.Python.

En la Figura 25 es veu un exemple on s'han especificat tots els tipus de test. Per cada tirador realitza tests simples en els temps [1.4, 1.7, 2.0] i tests dobles en [40, 45, 50]; un total de 12.

Després de processar la informació rebuda de l'Arduino, l'aplicació mostra els resultats de sí els temps compleixen l'estàndard de la FIE o no en unes taules específiques per cada modalitat. De manera més detalla s'indiquen els resultat de cada test individualment en un registre que es pot mostrar, ocultar o guardar en un arxiu de text si l'usuari així ho desitja.

4.4.2. Processament de dades

Quan l'usuari construeix un vector amb diferents valors de temps per cada test, l'aplicació s'encarrega d'enviar-los al microcontrolador utilitzant el PySerial. Per cada un dels valors s'envien de 3 a 4 bytes de manera simètrica a l'explicada en el capítol Funció mainS'informa del tipus de test, del tirador i s'envia el temps.

Per als tocats simples el nombre decimal introduït com a paràmetre es multiplica per 10 abans de ser enviat. D'aquesta manera s'evita treballar amb floats molt petits al Arduino i és més fàcil treballar amb enters en llenguatge C. El valor màxim que es pot arribar a

enviar segons l'aplicació és 290 (29ms), per tant, enviant un sol byte és suficient. Més tard, aquest valor es veurà multiplicat per 100 en l'Arduino per treballar en microsegons.

En el cas del tocat doble no s'ha de fer cap conversió, ja que el valor introduït no pot ser decimal. No obstant, el rang permès és entre 1-450 així que es divideix el número en 2 bytes abans de ser enviat. Totes aquestes mesures es tenen en compte per reduir al mínim el temps que s'estan comunicant l'aplicació i el microcontrolador.

Per cada un dels valors de temps, el programa envia i espera la resposta per donar temps al mòdul del tirador a reiniciar-se. La resposta de l'Arduino són 1's i 0's que indiquen si s'ha detectat o no respectivament, els tocats simples o dobles. Aquestes respostes es guarden en llistes que s'avaluen un cop s'ha finalitzat tots els tests d'un sol tirador, en una sola modalitat.

Tot seguit, amb aquesta llista de resultats, s'estudia el comportament del sistema sota test en l'interval de temps senyalat per buscar-hi possibles irregularitats. Suposant que el sistema no sempre indica un tocat vàlid per sobre d'un temps concret, i que pot variar la resposta quan es treballa amb increments molt petits de temps; es calcula l'interval entre els senyals que no s'han detectat i aquells que si ho han fet. Si aquest interval està dins de l'establert per la FIE, es considerarà dins de l'estàndard.

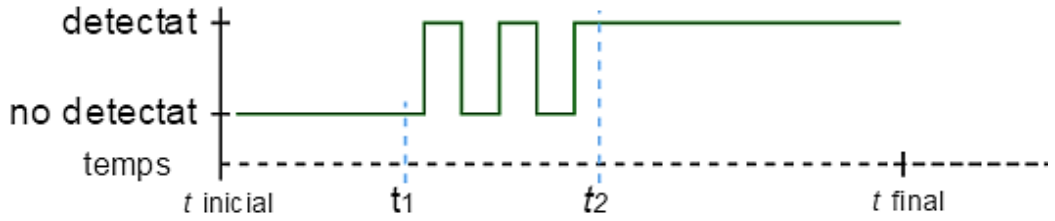

Figura 26. Zona de comportament irregular en el test dels tocats simples.

En la Figura 26 s'aprecia un exemple en el que el sistema es comporta de manera irregular per una simulació de tocats simples. Mentre que t1 estigui per sobre o sigui igual al mínim i t2 estigui per sobre o sigui igual al màxim, serà correcte. Amb els tocats dobles també es té en compte aquest comportament, però mentre que en els simples s'espera que a l'inici no es detectin els tocats (pels que són massa curts per considerar-se vàlids), en els dobles s'espera el contrari, és a dir, que al inici es capturin els tocats dobles fins arribar a cert punt en que només es detecta un dels dos tiradors.

En el moment de representar els resultats en les taules mostrades en la interfície gràfica, es té en compte si hi ha hagut un canvi entre capturats i no capturats, ja que el valor que es mostra a la taula és el temps llindar en que hi ha hagut aquest canvi i si aquest llindar entra en la normativa de la FIE o no.

Per aquells casos en que es realitza un sol test amb un únic temps, o tots els resultats de la simulació indiquen que s'han detectat o no els senyals, de manera que no es pot apreciar un llindar; el registre mostra per cada un temps si s'estan comportant correctament o no segons l'estàndard.

5. RESULTATS EXPERIMENTALS

En aquest apartat es mostren diferents resultats experimentals per demostrar el correcte funcionament dels components de la placa de circuit imprès fabricada així com el del sistema en general a través de les simulacions. Per aquestes simulacions s'ha utilitzat el Wireless Fencing produït en els projectes anteriors. En la Figura 27 s'aprecia el muntatge complet del sistema sota test juntament amb el Fencing Tester.

Figura 27. Wireless Fencing utilitzant Fencing Tester per fer simulacions.

En el moment de realitzar els test només s'han de tenir en compte un parell de coses. Primerament, que el mòdul que es configuri com a tirador vermell es connecti a través dels connectors del mateix color del Fencing Tester, i de manera semblant que el sensor LDR amb una etiqueta de color vermell s'encari amb el LED del mateix color. Amb aquests petits preparatius ja es pot obrir la aplicació i iniciar les simulacions.

5.1. Simulacions de tocats simples

En la Figura 28 es mostra la resposta dels relés al simular un tocat de 3ms. El senyal (1) en groc representa la sortida del Pin4 (tocat verd) i el senyal (2) en blau la sortida del relé. Aquest senyal apareix doblat i amb soroll ja que els mòduls dels tirador funcionen amb un senyal altern quadrat que pren dos nivells. No obstant, la part interessant és el retard del

senyal en passar el relé d'uns 80µs als que se li hauria de sumar els 240µs que tarda el senyal en estabilitzar-se a un voltatge suficient com per detectar el tocat. En total, el tocat tarda 320µs en enregistrar-se des de que surt del microcontrolador.

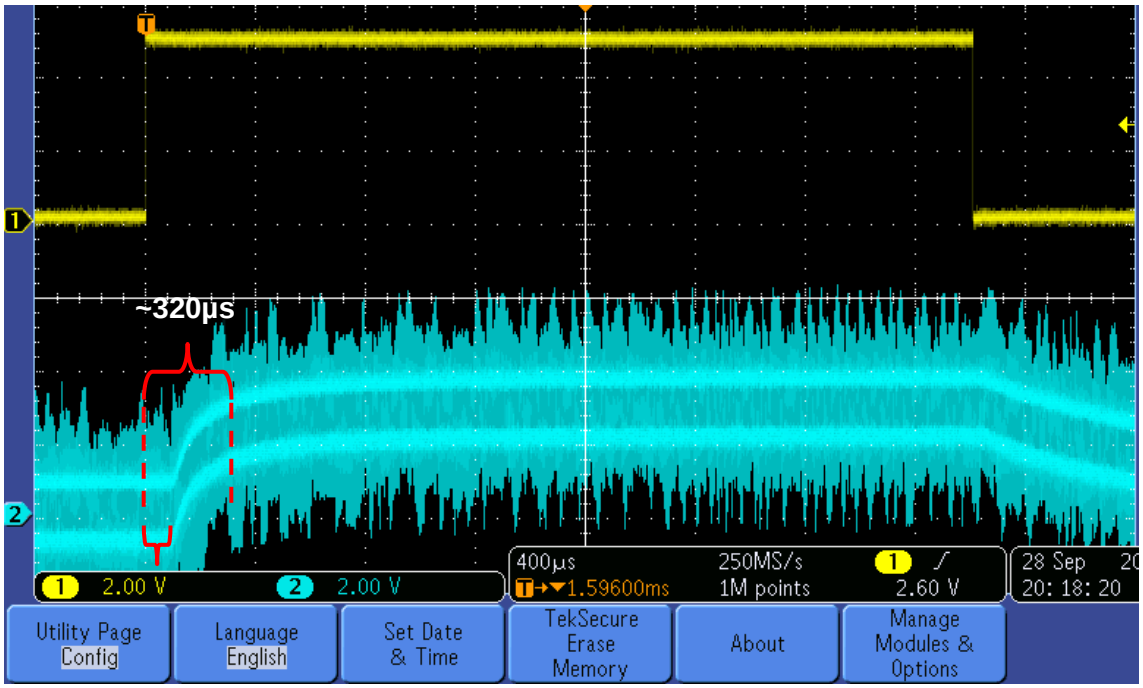

Figura 28. Retard relé estat sòlid amb tocat de 3ms.

Vist aquest valor, si es té en compte que els temps requerits per al tocat de sabre estan en un interval que comença als 100µs es té molt poc marge de maniobra. Es pot aconseguir sota la suposició que el relé es comporta sempre de la manera mostrada, no obstant, segons la fitxa d'especificacions el temps de resposta més ràpid del relé hauria de ser de 700µs. La conclusió més honesta és esperar que els tests de tocat simple per a la modalitat de sabre no donin resultats fiables al 100%, ja que el més probable és que per les simulacions d'una durada inferior a 100µs, no donin temps al relé a activar-se del tot.

La resposta del relé a l'aturar el tocat és pràcticament immediata, encara que el senyal tarda una estona més en arribar a 0V no és un temps que afecti a la interpretació del tocat.

En la següent prova es comprova el retard del senyal de tocat des que surt del microcontrolador fins que el sensor de llum enregistra el resultat en el mòdul central i, per tant també ho fa el microcontrolador. En aquesta ocasió, la simulació té 1.7ms de durada per al tocat representat de color groc, i la resposta del LED en color blau com es mostra en la Figura 29.

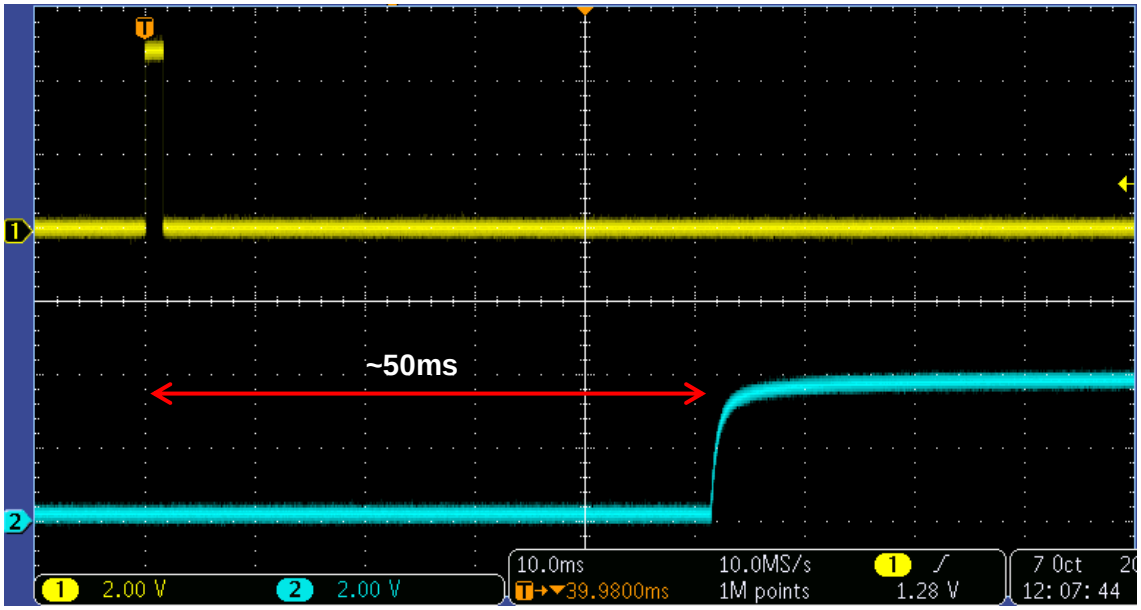

Figura 29. Tocat simple de 3ms i resposta LDR.

Tot i que el senyal de tocat dura només 1.7ms com s'observa al inici de la imatge, el LDR no enregistra la resposta fins 50ms més tard. Això ens confirma la suposició que s'havia fet inicialment al estudiar com es devia de comportar el sistema de tocats en el moment de calcular-los.

L'equipament que s'està utilitzant per fer aquestes proves, el Wireless Fencing està pensat solament per la modalitat d'espasa. Els requisits per la homologació en espasa especifiquen que un tocat ha de bloquejar l'altre durant 45 ± 5ms, així que el sistema, després de detectar el tocat d'un tirador, no encén el LED enseguida, s'espera almenys 50ms per comprovar que el segon tirador no realitzi també un tocat i només a partir d'aquest moment pot determinar si hi ha hagut un guanyador o ha sigut un empat.

Per tant, primer de tot, es confirma que el sensor de llum funciona correctament, capturant la llum del LED i enviant un senyal de 4V al microcontrolador. També es confirma que s'espera el temps correcte per determinar un doble tocat segons els requisits de la homologació. No obstant, el mòdul central està indicant un tocat correcte i segons la normativa de la FIE un tocat d'espasa només es pot considerar vàlid a partir dels 2ms mentre que el simulat només dura 1.7ms. Per tant, es pot concloure que el Wireless Fencing no pot passar l'homologació per als tocats simples mentre no arregli aquest petit error.

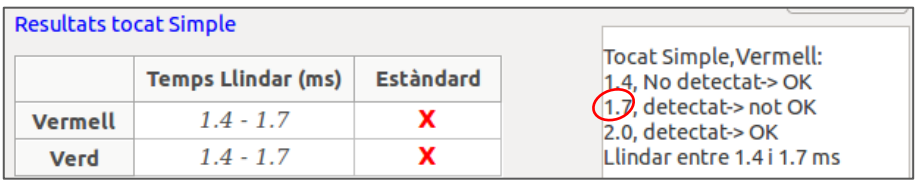

Figura 30. Resultats de test fallit en 1.7ms en la interfície gràfica

Si s'amplia la imatge del test realitzat per observar més detingudament el comportament del LDR s'observa un retard d'uns 600µs per al LED de color verd i tenint en compte que el microcontrolador captura el senyal a partir del 2,6V. Aquesta mesura no és molt important ja que un cop que el LED s'il·lumina ho fa durant el temps suficient per ser capturat, però en tot cas es verifica el bon funcionament dels components de la PCB.

5.2. Simulacions de tocats dobles

En la Figura 31 es mostren els resultats d'un test per a un tocat doble utilitzant el mateix sistema Wireless Fencing per espases. El senyal rosa representa el tirador vermell i el primer en fer el tocat, el senyal verd representa el segon tirador.

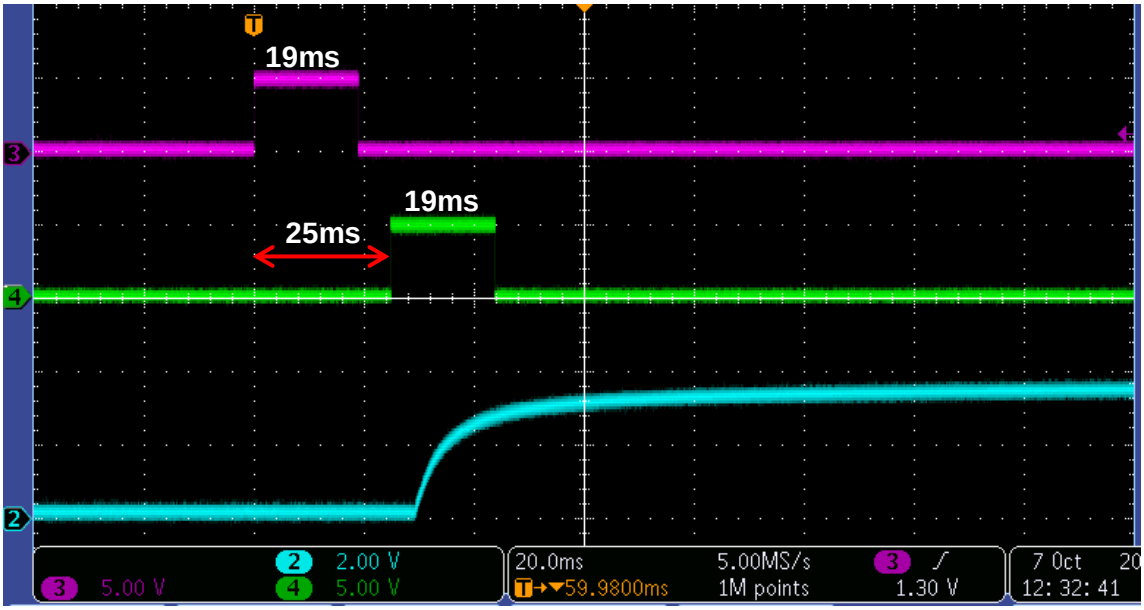

Figura 31. Tocat doble amb resposta LDR.

La simulació s'ha realitzat amb una diferencia de $t = 25ms$ entre tocats i, per tant, segons la normativa de la FIE significa que hauria de detectar un doble tocat ja que s'han realitzat en un interval inferior als 40ms. El senyal de color blau és la resposta del LDR llegint el LED vermell. El sistema sota test indica el doble tocat encenent els dos LED de colors simultàniament, i encara que no s'apreciï el LED verd a la gràfica, el resultat del test indica exitosament el doble tocat tal i com s'aprecia a la Figura 32.

Figura 32. Resultats de test doble correcte en la interfície gràfica

En aquesta ocasió s'observa com el sistema no espera 50ms com abans en il·luminar el LED. Això és degut a que en el moment que captura el segon tirador en l'interval de temps preestablert, ja no necessita comprovar cap altre possible senyal i per tant ja pot mostrar el resultat immediatament.

Vegem els resultats al realitzar el test amb una t superior als 50ms, concretament amb $t = 60ms$ per determinar si el sistema es comporta correctament.

En la Figura 33 només s'aprecia el resultat del LDR vermell, però la interfície gràfica indica que no hi ha hagut un doble tocat i per tant, només s'ha il·luminat el LED vermell per indicar el primer tirador com a guanyador. De nou, en la Figura 32 mostra l'aplicació mostrant el resultat correcte.

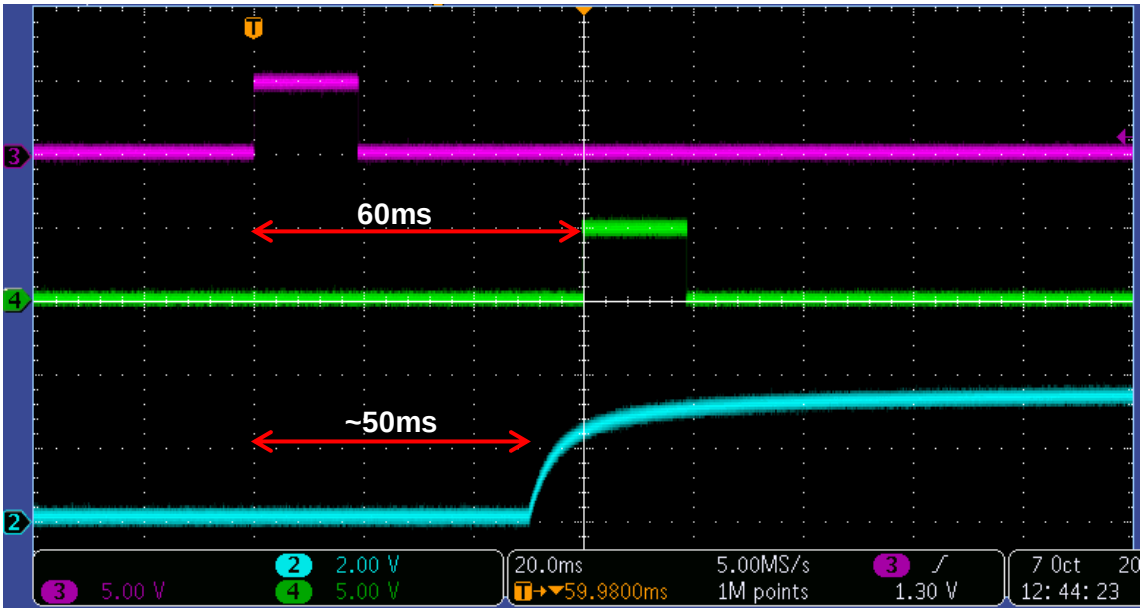

Figura 33. Tocat doble fora de l'interval de temps.

Observant la imatge es pot comprovar com, per una diferencia de temps superior als 50ms, el sistema ja no comprova si hi ha un segon tirador i per aquest motiu, precisament als 50ms il·lumina el LED vermell. És important recordar que aquests intervals esmentats són únics per la modalitat d'espasa que s'està testejant aquí.

Si bé el sistema Wireless Fencing no ha passat el test de tocats simples perquè indica tocats massa curts com a vàlids, el fet de senyalar tots aquells que superen els 10ms permet que se li puguin realitzar els test per als tocats dobles, que ha superat exitosament.

6. CONCLUSIONS

Durant la realització del projecte s'ha fet una recerca dels components que serien necessaris per aconseguir els objectius proposats i a partir d'ells s'ha dissenyat un circuit per dur-los a terme. El resultat d'això ha sigut un producte final amb un acabat compacte i robust presentable al públic i amb un cost molt econòmic.

Amb les eines disponibles s'ha aconseguit dissenyar un programa capaç de simular tocats simples pels tiradors amb diferents durades, així com tocats simultanis amb poques mil·lèsimes de segon de diferència per determinar-ne l'empat.

S'ha aconseguit proporcionar una interfície gràfica que permet als usuaris realitzar diferents simulacions estàndard o personalitzar-les escollint els paràmetres desitjats, tot plegat en una aplicació senzilla d'utilitzar.

Després de realitzar diverses proves amb el sistema de detecció de tocats del projecte de Ferreiro Pareja [1], s'ha pogut determinar que l'equipament en qüestió no supera els requisits de temps imposats per la Federació d'Esgrima per als tocats simples i que, per tant, s'hauria de retocar els seus paràmetres abans de poder passar la homologació.

Els resultats de les proves experimentals han posat de manifest que el sistema construït és robust i respon correctament a les necessitats i objectius proposats, no obstant és important remarcar que només s'ha pogut posar sota test equipament destinat únicament a la modalitat d'espasa i convindria disposar de més equipament per poder assegurar amb tota seguretat que funciona en tots els aspectes.

6.1. Línies futures

Tot i que el projecte ha resultat amb un producte final exitós, s'expliquen algunes modificacions o idees que es podrien aplicar per millorar-lo.

Durant la seva realització s'ha arribat a la conclusió que els càlculs del tocat simple per la modalitat de sabre s'acosten massa a les limitacions imposades pel hardware, així que una via de millora futura seria trobar una alternativa que reaccionés més ràpid al senyal enviat, mantenint igualment els sistemes elèctricament independents.

El sistema està pensat sobretot per l'equipament sense fills, tot i que calcula igualment els temps per als que funcionen amb cablejat. no obstant, per aquest no es tenen en compte els requisits sobre la validació de les impedàncies explicades al Capítol 2. Implicaria un redisseny important del sistema, però es podria afegir aquesta funcionalitat quan s'utilitzi equipament amb cablejat. Tot i que implicaria construir dos sistemes diferents, un per a equips sense fils i un per als que utilitzen cablejat, ja que funcionen de manera diferent.

Una altra via de millora podria incloure un mètode que fos capaç de determinar amb precisió el retard total del sistema sota test des que aquest reconeix un tocat com a vàlid, fins que es mostra la resposta. En els sistemes sense fils serviria per determinar si el retard es veu afectat per interferències en el canal de ràdio o per altres factors.

7. BIBLIOGRAFIA

[1] COSTA, Raúl Juan. *Ingeniería completa de un sistema wireless para la detección de tocados de esgrima: Sistemas microcontrolador y de comunicaciones*. Projecte final de carrera, UPC. Gener de 2009.

[2] COSTA, Raúl Juan. *Ingeniería completa de un sistema wireless para la detección de tocados en esgrima: Sensor*. Projecte final de carrera, UPC. Gener de 2009.

[3] SOLANS BATLLE, Daniel. *Disseny i optimització d'un sistema de detecció de tocats sense fils per a l'esport de l'esgrima: mòdul estàtic*. Projecte final de carrera, UPC. Març de 2011.

[4] SOLANS BATLLE, Daniel. *Protocolo de comunicaciones para un sistema de detección de tocados en esgrima basado en el estándar IEEE 802.15.4*. Projecte final de carrera, UPC. Maig de 2011.

[5] PLA POVEDA, Albert. *Disseny i optimització d'un sistema de detecció de tocats sense fils per a l'esport de l'esgrima: mòdul dinàmic*. Projecte final de carrera, UPC. Març de 2011.

[6] ROYO I PEÑA, Joan Pau. *Incorporación de un algoritmo de supervisión y otras mejoras a un sistema de detección de tocados inalámbrico para el deporte de la esgrima*. Projecte final de carrera, UPC. Juny de 2011.

[7] FERREIRO PAREJA, Sergio. *Protocolo de asignación dinàmica de canal para la comunicación de un sistema de esgrima inalámbrico basado en el estándar ieee 802.15.4*. Projecte final de carrera, UPC. 2016.

[8] Atmel, 2015. ATmega328. [en línia] San José, USA. Disponible a: <http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf> [Consultat 25 Agost 2016]

[9] *Hitmate fencing without wires.* [en línia] (2011) Disponible a: <http://www.hitmate.co.uk/epee_hitmate.php> [Consultat 13 Maig 2016]

[10] Favero Electronics. *WF1, Wireless Fencing apparatus,* [en línia] Disponible a: <http://www.favero.com/en2_fencing_sport_wireless_fencing_equipment_apparatus_for_epee_foil_without_wires_device_for_training_scoring_machine-246-17.html> [Consultat 13 Maig 2016]

[11] British Fencing 2015, *Rules for Competitions. Book 3. Material Rules*. [en línia] Disponible a: <http://britishfencing.com/uploads/files/book_m_-_20151209.pdf> [Consultat 25 Agost 2016]

[12] Wikipedia, 2016. Fencing. [en línia] (30 Juliol 2016) Disponible a: <https://en.wikipedia.org/wiki/Fencing#Competitive_fencing> [Consultat 3 Agost 2016]

[13] Avago Technologies, *ASSR-1218/1219 and 1228 Solid State Relay (Photo MOSFET)*. [en línia] United States Agost 2015. Disponible a: <http://www.avagotech.com/products/optocouplers/industrial-plastic/other/solid-state-relay/assr-1218-003e> [Consultat 1 Maig 2016]

[14] Silonex Inc, *NSL-19M51 Cermaic Package 102082 REV 4*. [en línia] Canada. Disponible a: <http://www.farnell.com/datasheets/77395.pdf> [Consultat 1 Maig 2016]

[15] Vishay Telefunken, *Silicon NPN Phototransistor BPX43 REV2*. [en línia] Heilbronn, Germany. Disponible a: <http://pdf.datasheetcatalog.com/datasheets/90/124847_DS.pdf> [Consultat 2 Maig 2016]

[16] Dunn Robin, wxPyWiki *Front Page*. [en línia] (30 Agost 2016). Disponible a: <https://wiki.wxpython.org/> [Consultat 2 Setembre 2016]

[17] Bodnar Jan, wxPython tutorial. [en línia] (6 Maig 2016). Disponible a: <http://zetcode.com/wxpython/> [Consultat 2 Setembre 2016]

[18] Premier Farnell plc. [en línia] (2016). Disponible a: <http://es.farnell.com/> [Consultat 15 Setembre 2016]

8. ANNEXOS

8.1. Annex 1: Codi microcontrolador

Main.c

```
#include <avr/interrupt.h>
#include <stdio.h>
#include <stdint.h> //inclueix stdint.h
#include <avr/io.h>
#include "serial_device.h" //communicacio UART
#include <math.h>

volatile uint16_t comptador=0, comptador_us=0, prova_ICR1; //volatile pk
les remenarem dins d'una ISR

volatile int tests_fets,isr_LED_RED_detectada=0,
isr_LED_GREEN_detectada=0;
volatile uint16_t count=0;


static int uart_putchar(char c, FILE *stream);
static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar,
NULL,_FDEV_SETUP_WRITE);
static int uart_putchar(char c, FILE *stream){
  if (c == '\n')
    uart_putchar('\r', stream);
  loop_until_bit_is_set(UCSR0A, UDRE0);
  UDR0 = c;
  return 0;
}

void timer_stop(void){
  TIMSK1 &= ~(1<<OCIE1A); //deshabilitem interrupcions timer1, timer1
compare match disabled
  TCCR1B &= ~((1<<CS10)|(1<<CS11)|(1<<CS12)); // stop timer clock ,
TCCR1B &= 0b11111000
}

void iniciar_timer(uint16_t pre){
  //Reiniciem contador timer1
  TCNT1H = 0;
  TCNT1L = 0;

  //Inicialitem la conta posant el prescaler desitjat
  switch(pre){
  case 8:  TCCR1B |= ((0<<CS12)|(1<<CS11)|(0<<CS10)); //Prescaler 8 clock
de 2 MHz, periode 0.5 microsegons
    break;
  case 64: TCCR1B |= ((0<<CS12)|(1<<CS11)|(1<<CS10)); //Prescaler 64
clock de 250 kHz, periode 4 microsegons
    break;
```

```
  case 256: TCCR1B |= ((1<<CS12)|(0<<CS11)|(0<<CS10)); //Prescaler 64
clock de 62.5 kHz, periode 16 microsegons
    break;
  case 1024: TCCR1B |= ((1<<CS12)|(0<<CS11)|(1<<CS10)); //Prescaler 1024
clock de 15625 Hz, periode 64 microsegons
    break;
  default: TCCR1B |= ((0<<CS12)|(1<<CS11)|(0<<CS10)); //64
    break;
  }
}

void config_ports(void){
  //Configurar OUTPUTS
  DDRD |= (1 << DDD4);     //PIN4 -> Sortida senyal /*GREEN*/
  PORTD |= (0 << PORTD4);
  DDRD |= (1 << DDD5);     //PIN5 -> Sortida senyal /*RED*/
  PORTD |= (0 << PORTD5);
  //Configurar OSCILOSCOPI(senyals per comprovar coses al sociloscopi)
  //DDRD |= (1 << DDD2);     //PIN2 -> Sortida senyal OSCILOSCOPI (per
mirar senyals)
  //PORTD |= (0 << PORTD2);

  //Configurar INPUTS
  DDRD &= ~(1 << DDD6);    //PIN6 -> Entrada senyal /*GREEN*/
  DDRD &= ~(1 << DDD7);    //PIN7 -> Entrada senyal /*RED*/

  //Habilitat interrupcions registre PCINT23..16

  PCICR |= (1 << PCIE2);     //Habilitem interrupcions per al registre
on estan els pins del 0 al 7
  PCMSK2 |= ((1 << PCINT22)|(1 << PCINT23)); //Habilitem interrupcions
ens pins 6 i 7 (INPUTS LDR)
  }

void config_timer(void){

  timer_stop();
  TCCR1A = 0; //posant bits WGM11 i WGM10 a 0.
  TCCR1B = 0; //amb aixo tb estem parant el timer de contar(prescaler=0)

  TCNT1 = 0; //Posem valor de timer a 0.  "TCNT1H i TCNT1L"
  OCR1A = 7; //amb prescaler: 8, son 3.5 microsegons (7+1=4us)

  TCCR1B |= (1<<WGM12);//activem mode CTC, TOP=OCR1A, WGM13..10 queda:
b0100
  TIMSK1 &= ~(1<<OCIE1A); //deshabilitem interrupcions timer1,ja les
activarem quan el volguem utiltizar
}

ISR(TIMER1_COMPA_vect){
  //Quan hagi contat fins al valor OCR1A entrara aqui
  count+=1;
}

ISR(PCINT2_vect){
//si canvia algun valor del registre PCINT23..16
//(isr nomes activats a pins 6 i 7, mirar config_ports();)
  if (bit_is_set(PIND, PIND7)){ //INPUT LDR RED
    isr_LED_RED_detectada = 1;
```

```
  }
  if (bit_is_set(PIND, PIND6)){ //INPUT LDR GREEN
    isr_LED_GREEN_detectada = 1;
  }
}


void delay_ms(uint16_t temps_ms){
  //Utilitzada primariament per esperar entre tocats
  //Valor maxim 4180 (en ms*), prescaler=1024, periode= 64us
  uint16_t contador=(temps_ms/64.0)*1000;
  OCR1A = 0xFFFF; //no s'arriba al valor maxim de FFFF, limiten parametre
funcio a 4180
                  //L'aparell es reinicia en 4s aproximadament.
                  //aquest contador no utilitza interrupcions
  iniciar_timer(1024);
  while(TCNT1 < contador){}
  timer_stop();
}


int tocat_simple(char tirador, uint16_t temps_tocat){  //temps_tocat en
microsegons
  int PORT_tirador = 0b00000000;        //PIND4 o 5
  int led_iluminat = 0;                 //1 o 0
  uint16_t valor_ocr1a = temps_tocat/4.0; //4us, periode utilitzat en
aquesta funcio

  if(tirador == 'R'){
      PORT_tirador = PORTD5; // = 0b00100000;
  }
  else if(tirador == 'G'){
    PORT_tirador = PORTD4;   // = 0b00010000;
  }

  OCR1A = valor_ocr1a-1;   //OCR1A = 749;  //inici->(749+1)*4us = 0,3 ms

  PORTD |= (1<<PORT_tirador);  //-------(simulant tocat)

  TIFR1 |= (1<<OCF1A);  //flag de interrupcio borrat al posar un 1 a
akest bit
  iniciar_timer(64);    //començem a contar (timer a 250 KHz), 4us
periode, 4us*10000=40k= 40 ms
  TIMSK1 |= (1<<OCIE1A);//habilitar interrupcio per comparacio amb timer

  sei();              //habilitacio interrupcions generals
  while (count == 0){}//deixem passar el temps, el relee creara un c.c
del temps indicat per OCR1A
  //quan compti fins a OCR1A entrara a ISR(TIMER1_COMPA_vect) i el count
pujara a 1.
  timer_stop();

  PORTD &= ~(1<<PORT_tirador);   //--------(aturant el tocat)

  delay_ms(350); //no s'encendra el LED enseguida, fara comprovacio de
doble tocat abans. li donem temps

  cli();          //deshabilitar interrupcions. ja s'hauria d'haver ences
LED
```

```
  delay_ms(3880); // aqui sencendran i apagaran LEDS pero les isr son
apagades i no ho capturaran

  if (tirador == 'R'){
    if (isr_LED_RED_detectada == 1){
      led_iluminat = 1;}
    else if (isr_LED_RED_detectada == 0){
      led_iluminat = 0;}
  }

  else if (tirador == 'G'){
    if (isr_LED_GREEN_detectada == 1){
      led_iluminat = 1;}
    else if (isr_LED_GREEN_detectada == 0){
      led_iluminat = 0;}
  }

  isr_LED_RED_detectada = 0;
  isr_LED_GREEN_detectada = 0;
  count = 0;    //el contador s'ha posat a 1 en la ISR(TIMERA_COMPA)
  return led_iluminat;
}

void timer_doble_tocat(void){
  //el timer es crida 3 vegades durant 1 test de tocat doble aixi que es
fa una funcio especifica
  TIFR1 |= (1<<OCF1A);  //posem a 0 el flag de interrupcio del timer1
(escribint un 1)pg139
  iniciar_timer(256);       //començem a contar (timer a 15625
Hz),periode 64us
  TIMSK1 |= (1<<OCIE1A);   //habilitar interrupcio per comparacio amb
timer
  //sei();                    //habilitem interrupcionsg generals
  while (count == 0){}        //deixem passar el temps
  timer_stop();         //int timer1 deshabilitades, timer s'atura
  count = 0;
}


int tocat_doble( char tirador, uint16_t temps_tocat){
  //Si hi ha tocat dels 2 players amb menys de 40ms de diferencia ->
doble tocat(espasa)
  //temps_tocat-> temps entre inici d'un tocat i el seguent
  int tirador1 = 0b00000000, tirador2 = 0b00000000;
  int doble_detectat = 0;     //0 -> no|1 -> si
  uint16_t valor_X = (temps_tocat/16.0)*1000; //prescaler 1024, 64us
periode

  if(tirador == 'R'){
    tirador1 = PORTD5;    // Red
    tirador2 = PORTD4;    // Green
  }
  else if(tirador == 'G'){
    tirador1 = PORTD4;
    tirador2 = PORTD5;
  }

  if (temps_tocat < 19){    // temps entre tocat es menor que la durada
d'un tocat
```

```
    uint16_t t_superposats = (19000/64.0)-valor_X;      //quan x<19ms

    OCR1A = valor_X-1;
    PORTD |= (1 << tirador1);  //------1er-tirador------(simulant tocat)-

    sei();                  //a partir d'aqui mirem si es leds s'encenen

    timer_doble_tocat();     //contem fins OCR1A i ens aturem

    OCR1A = t_superposats-1;    //temps que els dos tiradors estan fent
tocat
    PORTD |= (1 << tirador2);        //----segon-tirador---(simulant
tocat)--

    timer_doble_tocat();     //ens esperem els temps que toquen els dos
tiradors

    PORTD &= ~(1<<tirador1);   //---tirador-1------(parem el tocat)------
    OCR1A = valor_X-1;         //temps restant del segon tirador

    timer_doble_tocat();     //el segon tirador acaba amb el seu tocat

    PORTD &= ~(1<<tirador2);   //---tirador-2------(parem el tocat)------

    delay_ms(350);
    cli();          //a partir daqui els dos leds ja shaurien dhaver
ences o no
  }
  else if (temps_tocat >=19){   //temps entre tocats es >= a la durada
dels tocats (19ms)
    uint16_t t_entre_tocs = valor_X-(19000/16.0);     //quan x>19ms

    OCR1A = (19000/16)-1; //(295+1)*64us =18992 us ~19000 us = 19ms
    PORTD |= (1 << tirador1);  //------1er-tirador------(simulant tocat)-
    sei();                  //a partir d'aqui mirem si es leds s'encenen

    timer_doble_tocat();     //contem 19ms i ens aturem
    PORTD &= ~(1 << tirador1);   //-----tirador 1------------

    if(t_entre_tocs){             //Si el valor de x=19ms, el temps entre
tocats es 0, no fa res
      OCR1A = t_entre_tocs-1;    //temps que els dos tiradors estan fent
tocat
      timer_doble_tocat();     //ens esperem els temps entre els tocats
    }

    OCR1A = (19000/16.0)-1;         //temps restant del segon tirador
    PORTD |= (1 << tirador2);        //-----segon-tirador---(simulant
tocat)--------

    timer_doble_tocat();     //el segon tirador acaba amb el seu tocat

    PORTD &= ~(1 << tirador2);   //---tirador-2------(parem el tocat)----

    delay_ms(350);  //el temps maxim per al doble tocat en floret es de
325
    cli();          //a partir daqui els dos leds ja shaurien dhaver
ences o no
  }
```

```
  delay_ms(3880); //retard per reiniciar sistema

  if ((isr_LED_RED_detectada == 1) && (isr_LED_GREEN_detectada == 1)){
    doble_detectat = 1;
  }
  else {
    doble_detectat = 0;
  }
  isr_LED_GREEN_detectada = 0;     //reiniciem el valor abans de tornar a
fer el test
  isr_LED_RED_detectada = 0;
  //count = 0;  //ja s'ha posat a 0 en la funcio timer_coble_tocat()

  return doble_detectat;
}


void setup (void){
  cli();          //deshabilitar interrupcions globals
  config_ports(); //entrades/sortides i interrupcions
  config_timer(); //Mode CTC, OCR1A=7, reiniciar valor a 0
  serial_init();
  stdout = &mystdout; //utilitza modul serial_device, per habilitar
enviament port serie
}

int main(void){
  char tirador_color, tipus_test;
  int iluminat=0;
  uint16_t valor_tocat;
  setup();

  while(1){
    if (serial_can_read()){

      tipus_test = serial_get();     // 'S' o 'D'
      tirador_color = serial_get();  // 'R' o 'G'

      if (tipus_test == 'S'){
            valor_tocat = serial_get(); //si es simple amb un byte nhi
haura prou
            iluminat = tocat_simple(tirador_color, valor_tocat*100);
//valor_tocat ve en ms*10, aki els pasem a micro
      }
      else if (tipus_test == 'D'){
            valor_tocat = serial_get(); //es guardara en el byte menys
significatiu
      valor_tocat |= (serial_get() << 8);

            iluminat=tocat_doble(tirador_color, valor_tocat);
//valor_tocat en ms
      }
      printf("%d\n",iluminat);
    }
  }
}
```

8.2. Annex 2: Codi Python

Fencing.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import wx, os, serial, time, wx.grid
from wx.lib.masked import NumCtrl
from wx.lib.buttons import *

#Configuracio port serie Arduino
PORT = '/dev/ttyACM0'
baud = 9600
arduino = serial.Serial(PORT, baud)

#Temps per obrir port serie i reiniciar arduino
time.sleep(2)



class Taula(wx.grid.Grid):
    def __init__(self, parent, pos):
        wx.grid.Grid.__init__(self, parent, pos=pos, size= (333,83))

class Main_Frame(wx.Frame):
    """
    Aquesta classe nomes conte el 'panel' i afegeix els metodes per la
barra de menu
    """
    def __init__(self, parent, title):
        wx.Frame.__init__(self, parent, title=title, size=(700,650))

        # Creacio de la barra de Menu
        arxiumenu= wx.Menu()

        menusave = arxiumenu.Append(wx.NewId(), "Guardar"," Guarda el
registre en un fitxer de text")
        arxiumenu.AppendSeparator()
        menuabout = arxiumenu.Append(wx.NewId(), "Info"," Informacio
sobre aquesta aplicacio")
        arxiumenu.AppendSeparator()
        menuexit = arxiumenu.Append(wx.NewId(),"Tancar"," Tancar la
aplicacio")

        self.about = wx.MessageDialog( self, " Fencing Tester creat amb
wx.Python v3.0.1.1\n"
                                       " Autor: Marc Garcia
Colome\n"
                                       " Contacte:
mgc6363@gmail.com\n"
                                       " 2016 EPSEM - UPC",
                                    "Sobre el Fencing Tester",
wx.OK)
        self.sortirono = wx.MessageDialog( self, " Estas segur? \n",
```

```
                        "Tancar aplicacio", wx.YES_NO)

        # Creant la barra de menu on va el submenu Arxiu.
        menuBar = wx.MenuBar()
        menuBar.Append(arxiumenu,"&Arxiu")
        self.SetMenuBar(menuBar)

        self.panel = Main_Panel(self)
        self.dirname = ''

        self.Bind(wx.EVT_MENU, self.OnSave, menusave);
        self.Bind(wx.EVT_MENU, self.OnAbout, menuabout);
        self.Bind(wx.EVT_MENU, self.OnExit, menuexit);

    def OnSave(self,e):
        # Guarda el contingut del registre(en el panel) en un arxiu de
text
        dlg = wx.FileDialog(self, "Escull un fitxer", self.dirname, "",
"*.*",
                            wx.SAVE | wx.OVERWRITE_PROMPT)
        if dlg.ShowModal() == wx.ID_OK:
            # s'agafa el text del registre
            contingut = self.panel.logger.GetValue()

            # Obrir l'arxiu per escriure i tancar
            self.filename=dlg.GetFilename()
            self.dirname=dlg.GetDirectory()
            filehandle=open(os.path.join(self.dirname,
self.filename),'w')
            filehandle.write(contingut)
            filehandle.close()
        dlg.Destroy()

    def OnAbout(self,e):
        self.about.ShowModal()

    def OnExit(self,e):
        opcio = self.sortirono.ShowModal()
        if opcio == wx.ID_YES:
            arduino.close()
            self.Close(True)


class Main_Panel(wx.Panel):
    """
    Classe principal de tipus panel que conte tots els widgets mostrats
en la GUI.
    """
    def __init__(self, parent):
        wx.Panel.__init__(self, parent)

        # Una sortida de text -> Registre (o log)
        self.logger = wx.TextCtrl(self, pos=(430,310), size=(210,300),
                                  style=wx.TE_MULTILINE | wx.TE_READONLY)
        wx.StaticText(self, label="Registre:", pos=(430,280))

        # Grid per resultats de Test Simple
```

```
        wx.StaticLine(self,
pos=(30,280),size=(330,1),style=wx.LI_HORIZONTAL)
        wx.StaticLine(self,
pos=(30,281),size=(330,1),style=wx.LI_HORIZONTAL)
        t_simple_lbl=wx.StaticText(self, label="Resultats tocat Simple",
pos=(30,300))
        t_simple_lbl.SetForegroundColour(wx.BLUE)
        self.grid_simple = Taula(self, (30,330))
        self.grid_simple.CreateGrid(2,2)
        for col in range(2):
            for fila in range(2):
                self.grid_simple.SetReadOnly(col,fila,True)

self.grid_simple.SetDefaultCellAlignment(wx.ALIGN_CENTRE,wx.ALIGN_CENTRE)
        self.grid_simple.EnableDragGridSize(False)
        self.grid_simple.DisableDragColSize()
        self.grid_simple.DisableDragRowSize()

        self.grid_simple.SetColLabelValue(0,"Temps Llindar (ms)")
        self.grid_simple.SetColLabelValue(1,"Est%sndard" %
      u"\u00E0")
        self.grid_simple.SetRowLabelValue(0, "Vermell")
        self.grid_simple.SetRowLabelValue(1, "Verd")
        self.grid_simple.SetColSize(0, 150);
        self.grid_simple.SetColSize(1, 100);
        self.grid_simple.SetRowSize(0, 25);
        self.grid_simple.SetRowSize(1, 25);

        # Grid per resultats de Test Doble
        wx.StaticLine(self,
pos=(30,450),size=(330,1),style=wx.LI_HORIZONTAL)
        wx.StaticLine(self,
pos=(30,451),size=(330,1),style=wx.LI_HORIZONTAL)
        t_doble_lbl=wx.StaticText(self, label="Resultats tocat Doble",
pos=(30,470))
        t_doble_lbl.SetForegroundColour(wx.BLUE)
        self.grid_doble = Taula(self, (30, 500))
        self.grid_doble.CreateGrid(2,2)
        for col in range(2):
            for fila in range(2):
                self.grid_doble.SetReadOnly(col,fila,True)

self.grid_doble.SetDefaultCellAlignment(wx.ALIGN_CENTRE,wx.ALIGN_CENTRE)
        self.grid_doble.EnableDragGridSize(False)
        self.grid_doble.DisableDragColSize()
        self.grid_doble.DisableDragRowSize()

        self.grid_doble.SetColLabelValue(0,"Temps Llindar (ms)")
        self.grid_doble.SetColLabelValue(1,"Est%sndard" %  u"\u00E0")
        self.grid_doble.SetRowLabelValue(0, "Vermell")
        self.grid_doble.SetRowLabelValue(1, "Verd")
        self.grid_doble.SetColSize(0, 150);
        self.grid_doble.SetColSize(1, 100);
        self.grid_doble.SetRowSize(0, 25);
        self.grid_doble.SetRowSize(1, 25);


        # COMBOBOX MODALITAT
        Modalitat_list = ['Espasa', 'Sabre', 'Floret']
```

```
        wx.StaticText(self, label="Modalitat d'esgrima:", pos=(30,30))
        self.mod_sel = wx.ComboBox(self, value="Espasa",pos=(190, 25),
size=(95,-1),
                           choices=Modalitat_list,  style=wx.CB_READONLY)
        self.string_actual = 'Espasa' #s'utiliza com a flag per saber que
es comença amb espasa
        self.Bind(wx.EVT_COMBOBOX, self.OnComboSel, self.mod_sel)

        # SIMPLE TOCAT Parametres
        wx.StaticText(self,label="Tocat Simple", pos=(30,75))
        wx.StaticLine(self,
pos=(125,82),size=(270,1),style=wx.LI_HORIZONTAL)
        wx.StaticLine(self,
pos=(125,83),size=(270,1),style=wx.LI_HORIZONTAL)
        self.tirador_sim_r = wx.CheckBox(self, label="Vermell", pos=(50,
110))  #Simple RED
        self.tirador_sim_g = wx.CheckBox(self, label="Verd", pos=(50,
130))     #Simple GREEN
            # Valors Simple Test Input
        wx.StaticText(self, label=("Inici (ms)"), pos=(160,105))
        self.sim_inici = NumCtrl(self, value=2.0, pos=(160, 125),
size=(60,-1), min=0.1, max=25,
                                 fractionWidth=1,allowNegative=False,
limited=True, autoSize=False)
        wx.StaticText(self, label=("Salt (ms)"), pos=(240,105))
        self.sim_salt = NumCtrl(self, value=4.0, pos=(240, 125),
size=(60,-1), min=0, max=25,
                                fractionWidth=1,allowNegative=False,
limited=True, autoSize=False)
        wx.StaticText(self, label=("Final (ms)"), pos=(320,105))
        self.sim_final= NumCtrl(self, value=10.0, pos=(320, 125),
size=(60,-1), min=0.1, max=25,
                                fractionWidth=1,allowNegative=False,
limited=True, autoSize=False)


        #DOBLE TOCAT test
        wx.StaticText(self,label="Tocat Doble", pos=(30,180))
        wx.StaticLine(self,
pos=(125,190),size=(270,1),style=wx.LI_HORIZONTAL)
        wx.StaticLine(self,
pos=(125,189),size=(270,1),style=wx.LI_HORIZONTAL)
        self.tirador_dob_r = wx.CheckBox(self, label="Vermell", pos=(50,
215)) #Doble RED
        self.tirador_dob_g = wx.CheckBox(self, label="Verd", pos=(50,
235))    #Doble GREEN
            # Valors Doble Test Input
        wx.StaticText(self, label=("Inici (ms)"), pos=(160,210))
        self.dob_inici = NumCtrl(self, value=40, pos=(160, 230),
size=(60,-1), min=1, max=450,
                                 allowNegative=False,
limited=True, autoSize=False)
        wx.StaticText(self, label=("Salt (ms)"), pos=(240,210))
        self.dob_salt = NumCtrl(self, value=5, pos=(240, 230), size=(60,-
1), min=0, max=35,
                                 allowNegative=False,
limited=True, autoSize=False)
        lbl_final_d = wx.StaticText(self, label=("Final (ms)"),
pos=(320,210))
```

```
        self.dob_final = NumCtrl(self, value=50, pos=(320, 230),
size=(60,-1), min=1, max=450,
                                         allowNegative=False,
limited=True, autoSize=False)

        #Logotip de tiradors d'esgrima
        self.png = wx.Bitmap("Logo_petit.png", wx.BITMAP_TYPE_ANY)
        wx.StaticBitmap(self, -1, self.png, pos=(445,25))

        # Boto REALITZAR TEST
        start_icon = wx.Bitmap("Start_little.png", wx.BITMAP_TYPE_ANY)
        GenBitmapTextButton(self, bitmap=start_icon,label="  Realitar
Test",
                            pos=(457, 145), size=(165,55))
        self.Bind(wx.EVT_BUTTON, self.OnButtonTest)

        # Boto REINICIAR PARAMETRES
        self.button_reiniciar =wx.Button(self, label=("Reiniciar
Par%smetres" % u"\u00E0" )
                                         , pos=(465, 210),style=1 )
        self.Bind(wx.EVT_BUTTON,
self.OnButtonReiniciar,self.button_reiniciar)

        # Boto MOSTRAR/OCULTAR logger
        self.button_showhide =wx.Button(self, label="Ocultar", pos=(555,
274))
        self.Bind(wx.EVT_BUTTON, self.OnButtonHide, self.button_showhide)
        self.estat_logger = True


    def OnButtonTest(self, e):
    # Event per capturar parametres de tests i enviar-los per fer els
tests
        comprova = self.CheckErrors()
        sel = False
        if comprova:
            l_sim = []
            l_dob = []
            #si esta seleccionat un tirador omplira la llista, sino es
quedara buida
            if (self.tirador_sim_r.GetValue() |
self.tirador_sim_g.GetValue()):
                sel=True
                if self.tirador_sim_r.GetValue():
                    l_sim.append('R')
                else:
                    l_sim.append(0)
                if self.tirador_sim_g.GetValue():
                    l_sim.append('G')
                else:
                    l_sim.append(0)
                i_s=self.sim_inici.GetValue()
                s_s=self.sim_salt.GetValue()
                f_s=self.sim_final.GetValue()
                l_sim = prepara_num_list(l_sim,i_s,s_s,f_s,'S')
            if (self.tirador_dob_r.GetValue() |
self.tirador_dob_g.GetValue()):
                sel = True
                if self.tirador_dob_r.GetValue():
```

```
                    l_dob.append('R')
                else:
                    l_dob.append(0)
                if self.tirador_dob_g.GetValue():
                    l_dob.append('G')
                else:
                    l_dob.append(0)
                i_d=self.dob_inici.GetValue()
                s_d=self.dob_salt.GetValue()
                f_d=self.dob_final.GetValue()
                l_dob = prepara_num_list(l_dob,i_d,s_d,f_d,'D')

            if not (sel):
                pass
            else:
                modalitat=self.mod_sel.GetValue() #combobox
                lists=[l_sim,l_dob]
                envia_tipus(lists,modalitat,self.logger,
self.grid_simple, self.grid_doble)

    def OnComboSel(self,e):
        """
        Si es canvia la modalitat, reinicia parametres i neteja grids.
        Si es selecciona la mateixa modalitat, no fa res.
        """
        flag = False
        if (e.GetString() == 'Espasa') &
(e.GetString()!=self.string_actual):
            flag = True
            self.string_actual = 'Espasa'
            self.sim_inici.SetValue(2.0)
            self.sim_salt.SetValue(4.0)
            self.sim_final.SetValue(10.0)
            self.dob_inici.SetValue(40)
            self.dob_salt.SetValue(5)
            self.dob_final.SetValue(50)
        elif (e.GetString() == 'Sabre') &
(e.GetString()!=self.string_actual):
            flag = True
            self.string_actual = 'Sabre'
            self.sim_inici.SetValue(0.1)
            self.sim_salt.SetValue(0.4)
            self.sim_final.SetValue(1.0)
            self.dob_inici.SetValue(160)
            self.dob_salt.SetValue(10)
            self.dob_final.SetValue(180)
        elif (e.GetString() == 'Floret') &
(e.GetString()!=self.string_actual):
            flag = True
            self.string_actual = 'Floret'
            self.sim_inici.SetValue(13.0)
            self.sim_inici.SetValue(0.5)
            self.sim_inici.SetValue(15.0)
            self.dob_inici.SetValue(275)
            self.dob_salt.SetValue(25)
            self.dob_final.SetValue(300)

        if flag:
            for col in range(2):
```

```
                for fila in range(2):
                    self.grid_simple.SetCellValue(col,fila,"")
            for col in range(2):
                for fila in range(2):
                    self.grid_doble.SetCellValue(col,fila,"")

    def CheckErrors(self):
        """
        Comprova que el valor final no es inferior al inicial i que hi
hagi un salt en cas de multiples tests
        """
        if (self.tirador_sim_r.GetValue() |
self.tirador_sim_g.GetValue()):
            if self.sim_final.GetValue() < self.sim_inici.GetValue():
                miss_error= wx.MessageDialog(None, 'El valor final no pot
ser inferior al inicial.',
                                             'Error en Tocat Simple',
wx.OK)
                miss_error.ShowModal()
                return False
            if (self.sim_final.GetValue() > self.sim_inici.GetValue()) &
(self.sim_salt.GetValue()==0):
                miss_error= wx.MessageDialog(None, 'No es pot arribar al
valor final sense un salt',
                                             'Error en Tocat Simple',
wx.OK)
                miss_error.ShowModal()
                return False
        if (self.tirador_dob_r.GetValue() |
self.tirador_dob_g.GetValue()):
            if self.dob_final.GetValue() < self.dob_inici.GetValue():
                miss_error= wx.MessageDialog(None, 'El valor final no pot
ser inferior al inicial.',
                                             'Error en Tocat Doble',
wx.OK)
                miss_error.ShowModal()
                return False
            if (self.dob_final.GetValue() > self.dob_inici.GetValue()) &
(self.dob_salt.GetValue()==0):
                miss_error= wx.MessageDialog(None, 'No es pot arribar al
valor final sense un salt',
                                             'Error en Tocat Doble',
wx.OK)
                miss_error.ShowModal()
                return False

        return True

    def OnButtonHide(self,e):
        #Amaga/Oculta el logger i canvia text del boto
        if self.estat_logger:
            self.logger.Hide()
            self.estat_logger = False
            self.button_showhide.SetLabel('Mostrar')
        else:
            self.logger.Show()
            self.estat_logger = True
            self.button_showhide.SetLabel('Ocultar')
```

```
    def OnButtonReiniciar(self, boto_r):
        pass


def prepara_num_list(llista,i,s,f,tipus):
    """
    Afegeix a llistes(simple i doble) els numeros preparats per ser
enviats a Arduino
    Simple-> ms*10 ,   Doble-> ms
    """
    dif=f-i
    if dif !=0:
        q_salts=dif/s
    else:  #quan nomes es vol fer un test i no hi ha salt
        q_salts=0
    for num in range(int(q_salts+1)): #si arriba a final be, sino no el
supera
        if tipus=='S':
            llista.append(int(i*10))
        else:   #doble en ms, no fa falta canviar
            llista.append(i)
        i+=s
    return llista

def envia_tipus(dues_l, mod, log,gs,gd):
    """
    Desde aqui s'envien les llistes sequencialment. Primer els Simples,
despres els Dobles
    """
    for i,list in enumerate(dues_l):          #l_sim i l_dob, plenes o
buides
        if len(list):       #mentre la llista estigui plena
            if i==0:
                envia_num_ard(list, 'S', mod, log,gs,gd)
            else:
                envia_num_ard(list, 'D', mod, log,gs,gd)


def envia_num_ard(nums,tipus,mod,log,gs,gd):
    """
    Avisa a arduino del test(S,D), tirador(R,G) i envia els valors pels
test de 1 en 1.
    Simple->nomes envia un byte (numeros >256)
    Doble ->envia dos bytes (num max que s'ennvia es >350 per floret)
    """
    for color in nums[0:2]:
        if color:
            if tipus =='S':
                log.AppendText('\nTocat Simple, ')
            else:
                log.AppendText('\nTocat Doble, ')
            if color == 'R':
                log.AppendText('Vermell:\n')
            else:
                log.AppendText('Verd:\n')
            l_resultat=[]
            for num in nums[2:]:
            #envia numero a arduino, i espera resposta que es guarda a
l_resultat
```

```
                if tipus=='S':
                    arduino.write(tipus)  #S o D
                    arduino.write(color)  #R o G
                    arduino.write(chr(num))
                    time.sleep(0.3)
                elif tipus=='D':
                    #num maxim per doble=450, necessari 2 bytes
                    if num > 255:
                        low_byte = '0b'+ bin(num)[-8:len(bin(num))]
                        low_byte = int(low_byte,2)
                        big_byte = bin(num)[0:len(bin(num))-8]
                        big_byte = int(big_byte,2)
                    else:
                        low_byte = num
                        big_byte = 0

                    arduino.write(tipus) #S o D
                    arduino.write(color) #R o G
                    arduino.write(chr(low_byte))
                    arduino.write(chr(big_byte))
                    time.sleep(0.3)
                resposta=arduino.readline()
                l_resultat.append(int(resposta[0]))   #separem '\n' i
convertim a int
                #els resultats s'envien quan han acabat tots els tests
dun tirador

processa_resultats(mod,tipus,color,l_resultat,nums[2:],log,gs,gd)
#l_resultats i nums[2:0] haurien de fer parella


def processa_resultats(mod,tipus,color,resultats,enviats,log,gs,gd):
    """
    Determina el llindar de temps en que hi ha hagut un canvi en
'deteccio->no deteccio'
    """
    if tipus=='S':
    #resultats de simple: s'espera que primers resultats siguin 0 i
ultims 1's
        for i,e in enumerate(enviats):
            #abans d'enviarlos els multipliquem x10, aqui els tornem a
dividir
            enviats[i]=e/10.0

        t1=0  #ultim temps en el que els tests no han ences el led
        t2=0  #sera el primer temps que ha aconseguit encendre el LED
        t_aux=0  #per quan hi ha comportament erratic
        for i, numero in enumerate(enviats): #amb i iterarem resultats
            if resultats[i]==0:
                if t2==0:
                    t1=numero
                else:
                    t_aux=numero
            elif resultats[i]==1:
                if (t2==0) & (t1!=0):
                    t2=numero
                elif (t2!=0) & (t_aux!=0):
                    t2=numero
                    t_aux=0
```

```
    elif tipus=='D':
    #resultats de doble: s'espera que els primers resultats siguin 1's i
ultims 0
        t1=0  #ultim temps en que s'ha detectat doble
        t2=0  #sera el primer temps que no ha detectat doble
        t_aux=0  #per quan hi ha comportament erratic
        for i, numero in enumerate(enviats): #amb i iterarem resultats
            if resultats[i]==1:
                if t2==0:
                    t1=numero
                else:
                    t_aux=numero
            elif resultats[i]==0:
                    if t2==0:
                        t2=numero
                    elif (t2!=0) & (t_aux!=0):
                        t2=numero
                        t_aux=0

    if mod[0]=='E':
        calcul_output('E',tipus,color,resultats, enviats,t1,t2,log,gs,gd)
    elif mod[0]=='S':
        calcul_output('S', tipus,color,resultats,
enviats,t1,t2,log,gs,gd)
    elif mod[0] == 'F':
        calcul_output('F',tipus,color,resultats, enviats,t1,t2,log,gs,gd)

def calcul_output(mod,tipus,tirador,results, enviats,t1,t2, log,gs,gd):
    """
    Calcula el temps llindar si existeix un. Contrasta resultats amb
homologacio FIE.
    Mostra resultats per cada test individual.
    """
    #Depenent de la modalitat els valors a tenir en compte seran
diferents
    if mod =='E':
        t_max_S = 10
        t_min_S = 2
        t_max_D = 50
        t_min_D = 40
    elif mod =='S':
        t_max_S = 1
        t_min_S = 0.1
        t_max_D = 180
        t_min_D = 160
    elif mod == 'F':
        t_max_S = 15
        t_min_S = 13
        t_max_D = 325
        t_min_D = 275

    result_test=[]
    if tipus =='S':
        #Creacio de outputs a log, individuals per cada tocat
        for i,num in enumerate(enviats):
            if (results[i] == 1):
                if (num >= t_min_S):
                    log.AppendText('%s, detectat-> OK\n' % num)
```

```
                else:
                    log.AppendText('%s, detectat-> no OK\n' % num)
            else:
                if (num < t_max_S):
                    log.AppendText('%s, No detectat-> OK\n' % num)
                else:
                    log.AppendText('%s, No detectat-> no OK\n' % num)

        #Calcul del llindar, si n'hi ha
        if (t2 >= t_min_S):
            if (t1<t_max_S):
                passa_test = True
            else:
                passa_test = False
        else:
            passa_test = False
        t_llindar = (t1,t2)
        result_test = ['S',tirador,t_llindar, passa_test]

        write_grid(result_test,gs,gd,log)

    elif tipus=='D':
        #Creem outputs a log, individuals per cada tocat
        for i,num in enumerate(enviats):
            if (results[i] == 1):
                if (num < t_max_D):
                    log.AppendText('%s, detectat-> OK\n' % num)
                else:
                    log.AppendText('%s, detectat-> no OK\n' % num)
            else:
                if (num >= t_min_D):
                    log.AppendText('%s, No detectat-> OK\n' % num)
                else:
                    log.AppendText('%s, No detectat-> no OK\n' % num)

        if (t2 >= t_min_D):  #valors en ms
            if (t1 < t_max_D):
                passa_test=True
            else:
                passa_test=False
        else:
            passa_test=False
        t_llindar=(t1,t2)
        result_test = ['D',tirador,t_llindar, passa_test]

        write_grid(result_test,gs,gd,log)

def write_grid(result, grid_s, grid_d, log):
    """
    Mostra resultats en el grid
    """
    flag=False
    t1 = result[2][0]
    t2 = result[2][1]

    if result[0] == 'S':
        grid = grid_s
    elif result[0] == 'D':
        grid = grid_d
```

```
    if result[1] == 'R':
        fila = 0
    elif result[1] == 'G':
        fila = 1

    if (t1!=0) & (t2!=0):
        grid.SetCellValue(fila, 0,"%s - %s" % (t1, t2))
        log.AppendText('Llindar entre %s i %s ms\n' % (t1, t2))
        flag = True
    elif (t1 == 0) | (t2 == 0):
        #si no hi ha un llindar no es pot considerar que pasi un
estandard
        grid.SetCellValue(fila, 0,"sense llindar")
    grid.SetCellFont(fila, 0,wx.Font(11,wx.ROMAN,wx.ITALIC,wx.NORMAL))

    #Dibuix de l caracter 'check' o la creu
    if result[3] & flag:
        grid.SetCellValue(fila,1,"%s" % u"\u2713")
        grid.SetCellTextColour(fila,1,(0,102,0))
        grid.SetCellFont(fila,1,wx.Font(19,wx.DEFAULT,wx.NORMAL,wx.BOLD))
    elif (result[3]==False) & flag:
        grid.SetCellValue(fila,1,"X")
        grid.SetCellTextColour(fila,1,wx.RED)
        grid.SetCellFont(fila,1,wx.Font(12,wx.SWISS,wx.NORMAL,wx.BOLD))
    else: #si no hi ha un llindar no escriu res
        grid.SetCellValue(fila,1,"")


app = wx.App(False)
frame = Main_Frame(None, "Fencing Tester")
frame.Center()
frame.Show()
app.MainLoop()
```